# NETWORK-BASED VISUAL ANALYSIS OF TABULAR DATA

A Dissertation
Presented to
The Academic Faculty

by

Zhicheng Liu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology
May 2012

# NETWORK-BASED VISUAL ANALYSIS OF TABULAR DATA

Approved by:


Dr. John Stasko, Advisor
School of Interactive Computing
*Georgia Institute of Technology*

Dr. Eric Gilbert
School of Interactive Computing
*Georgia Institute of Technology*

Dr. James Foley
School of Interactive Computing
*Georgia Institute of Technology*

Dr. Shamkant Navathe
School of Computer Science
*Georgia Institute of Technology*

Dr. Haesun Park
School of Computational Science and Engineering
*Georgia Institute of Technology*

Dr. Jarke van Wijk
Department of Mathematics and Computer Science
*Eindhoven University of Technology*



Date Approved: March 30th, 2012

*To my family, especially my parents.*

# ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor John Stasko. My intellectual growth and academic accomplishments over the past five years are simply not possible without him. An ideal mentor one could ask for, he has provided continued support and guidance, pushed me to challenge my limits, and given me freedom to explore things that truly excite me. He truly cares about his students, and his advice and teachings will continue to benefit me immensely in the years to come.

I would also like to thank my dissertation committee members, Eric Gilbert, James Foley, Shamkant Navathe, Haesun Park and Jarke van Wijk. Their insightful and incisive feedback made this dissertation much better. It was especially a pleasure to work closely with Sham Navathe, whose expertise and experience in database theory is extremely valuable. I am also in deep gratitude for the advice and help on career given by John and the committee members.

Another person who had a great influence on my graduate career is Nancy Nersessian. There is a part of me that strives to be critical and even philosophical in thinking about the relationship between human and computing technology, and Nancy has clearly shaped my theoretical orientation in this regard. I thank her very much for the delightful and eye-opening discussions in her class and in her office.

Finally, many thanks to my colleagues in the Information Interfaces Lab and the Human Centered Computing program, and my friends in Georgia Tech, for making the graduate school a more enjoyable experience.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

Tabular data is pervasive in the form of spreadsheets and relational databases. Although tables often describe multivariate data without explicit network semantics, it may be advantageous to explore the data modeled as a graph or network for analysis. Even when a given table design conveys some static network semantics, analysts may want to look at multiple networks from different perspectives, at different levels of abstraction, and with different edge semantics.

This dissertation is motivated by the observation that a general approach for performing multi-dimensional and multi-level network-based visual analysis on multivariate tabular data is necessary. We present a formal framework based on the relational data model that systematically specifies the construction and transformation of graphs from relational data tables. In the framework, a set of relational operators provide the basis for rich expressive power for network modeling.

Powered by this relational algebraic framework, we design and implement a visual analytics system called Ploceus. Ploceus supports flexible construction and transformation of networks through a direct manipulation interface, and integrates dynamic network manipulation with visual exploration for a seamless analytic experience.

# CHAPTER I

# INTRODUCTION

## *1.1 Motivation*

With the advances in digital sensors, communications, computation, and storage, it is now easier than ever for individual consumers, business corporations and government agencies to generate and store massive amounts of data. Analyzing and making sense of data and turning the data into insights to drive decision making and scientific discovery have become one of the greatest challenges and opportunities of the 21st century [51, 88].

A significant portion of the data in today's world is stored in the form of structured tables, as a result of the dominance of technologies such as spreadsheets and relational databases in current data management practices. It is a common convention to represent data cases using table rows, and to represent attributes or dimensions of the data cases using columns.

To make sense of such a multidimensional dataset, the first step is typically exploratory data analysis [90]: to look at the data to see what it seems to tell us. The goal of exploratory data analysis is thus about *description*, not confirmation. Subsequently, when we discover something interesting, tools like statistics are useful to perform confirmation analysis.

Visualization is an effective tool for exploratory analysis. As John Tukey put it, "the great value of a picture is when it forces us to notice what we never expected to see" [90]. Previous research provides a rich collection of approaches to visually explore multidimensional tabular data including the Table Lens [76], InfoZoom [80], Polaris [86], Taleau [6], to name just a few.

These systems support visualization techniques such as a line graph, (stacked) bar chart, scatter plot, heatmap and geographical maps (Figure 1). These visualizations are effective to help analysts discover patterns and outliers in terms of aggregation and correlation of quantitative or count data. We can thus easily answer questions like "How are my company's sales numbers trending this year?" or 'What is the distribution of low-income families like in this area?" with these visualizations. With the exception of maps, Bertin categorizes these techniques in his seminal book *Semiology of Graphics* [19] as diagrams.

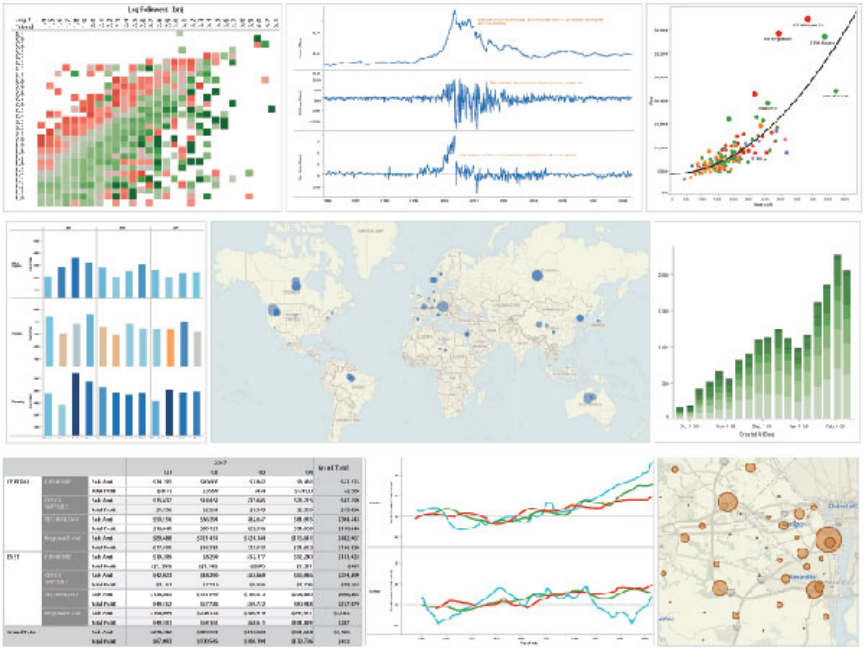


**Figure 1:** A dashboard style interface in Tableau showcasing different visualizations supported in the system

Another type of visualization in Bertin's taxonomy is the network. A network is a commonly used conceptual tool to understand the complex interaction between people, objects, processes and events in the real world. Typically a network is depicted using a node-link visualization, but other variants such as a matrix-based representation are possible too (Figure 2). Compared with diagrams, networks are good for exploring and answering different kinds of questions. For example, how are people's social relationships manifested in the form of clusters? Who is the gate keeper that

connects two disjoint communities? Diagrams are not well suited to answer these questions.

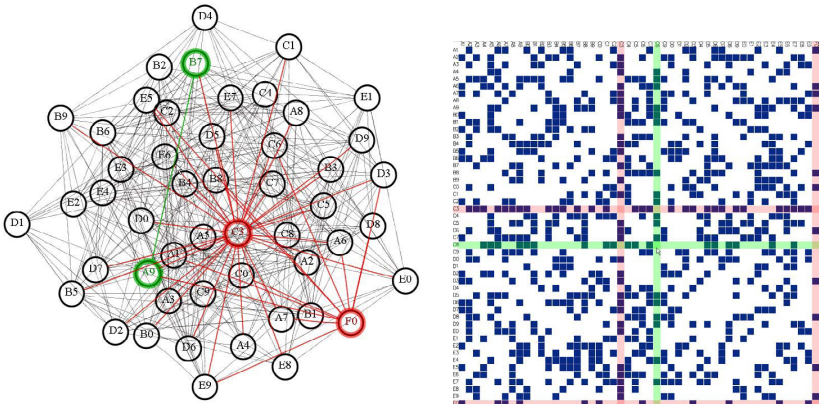


**Figure 2:** Network visualizations of the same graph in node-link and matrix representations [41].

To conduct network-based exploratory analysis, we must first have a network data structure. Analysts thus need to declare and model explicit relationships between entities in the data collection and formatting stage. Network visualizations, often in the form of node-link diagrams, and other tools such as statistics are then used to analyze patterns of interaction between entities, to discover entities with interesting roles, and to identify inherent groups or clusters of entities.

Systems such as Table Lens and Tableau do not provide support for network modeling on tabular data. As a result, network visualizations are absent as a data exploration tool from these systems. On the other hand, existing approaches to visual analysis of networks (e.g. UCINet [7], NodeTrix [50]) usually assume a given graph and static network semantics.

Consequently, to perform network-based visual analysis on tabular data, analysts must first anticipate which data cases and attributes should be included for network analysis, write custom code to extract a network from data tables, represent the network data using GraphML [2] or other related file formats, and then feed the data into network analysis systems. Systems like NodeXL [79, 47] try to leverage

the interface of Microsoft Excel to avoid the use of a programming language for data manipulation, but the analytical power of Excel has fundamental limitations due to a lack of well-defined table algebra and the support for flexible and expressive queries [63, 99].

The current state of the art thus entails two fundamental limitations for performing network-based visualization on tabular data. First, support for dynamic network modeling is inadequate. During an analysis process, selecting, filtering, clustering or computing metrics over a static network is not always enough. Analysts may want to construct new networks and transform existing ones to explore the data from different perspectives and at different levels of abstraction. The requirement of technical skills to model these networks results in a significant overhead for analysts. The often tedious modeling process constrains analysts in choosing what to see and connect and exploring multiple related networks with different semantics.

Second, data modeling and visual analysis constitute disjoint processes. Analysts can not get direct and immediate visual feedback during the process of data manipulation. The entire work flow is in contradiction with the serendipity of exploratory analysis, where new tasks and questions can emerge during the process of data exploration.

## *1.2 Objective*

To address these limitations, the overarching research question of this dissertation is therefore:

> *Given the current pervasiveness of tabular data, is it possible to design and build a system that supports flexible user-initiated network modeling from raw data tables, reduces semantic distance in answering analytic questions, and helps to create a holistic exploratory analysis experience?*

We do not intend to restrict our investigation within specific problem domains. Instead, we seek to provide a general solution that can be applied to tabular data in a variety of domains. It is, however, impossible to know and capture all the network semantics that analysts want to articulate given the diversity of datasets and analytical tasks. Even in a specific problem domain, new semantics may only emerge during the process of exploratory analysis. As a result, it is a great challenge to provide a general system that can potentially support a wide range of datasets and tasks.

Our approach is formalism-based. Inspired by the relational model underlying database theory [28] and other formalism-based approaches to visualization and analytics [84, 98], we break down the question into two interrelated problems.

First, which operations are necessary to extract and transform tabular data into networks or graphs? Formulated formally:

Given a tabular data $R$, transform $R$ into a graph $G$

$G = \{N, E\}, N = f_{node}(R)$ & $E = f_{edge}(R)$

where $f_{node}$ and $f_{edge}$ are user-defined functions that map $R$ to a node set $N$ and an edge set $E$ respectively.

The first part of this dissertation thus explores the possible $f_{node}$ and $f_{edge}$ functions, organizes them into a coherent framework, and provides consistent and tractable mechanisms for computing these functions.

Secondly, given a set of formal operations, how do we design a system that incorporates all these operations, enables intuitive access to these operations, and tightly couples network modeling with exploratory analysis?

The second part of this dissertation addresses this design problem by adopting the following design principles: using direct manipulation to reduce articulatory distance, integrating network modeling with visual exploration for seamless experience, and providing visualization management support.

## 1.3 Contributions

The contributions of this dissertation include:

- A conceptual framework specifying possible operations for constructing and transforming networks from multivariate tabular data.
- A specification of the conceptual operations based on the relational model and an implementation of the framework in relational algebra.
- The design and implementation of a system, Ploceus, based on the framework that supports flexible exploratory analysis. Semantics articulation, graph generation and visual exploration are tightly integrated.
- A discussion of the nature of high-level tasks in network-based visual analysis that have potential implications for future work on visual analytics.

## 1.4 Organization

The rest of this dissertation is organized as follows.

In Chapter 2, we provide an overview of different types of tabular data and graphs. We then define clearly the scope of our research: the types of tabular data we plan to investigate, and the types of graphs that our approach supports. We conclude the chapter with a discussion of the nature of user tasks in network-based visual analysis, and its implication on system design.

In Chapter 3, we give a detailed literature review of related work. Since the research in this dissertation is at the intersection of network analysis and tabular data analysis, we organize this chapter by discussing the visual methods and computational/statistical methods for analyzing networks and data tables respectively. We then survey hybrid approaches that investigate potential connections between networks and data tables. In reviewing the related work, we discuss how our research fits in

a larger picture of multidimensional data analytics and what potential contribution our research makes.

Chapter 4 presents a formal framework as a solution to the first research question highlighted in this introduction: what kind of operations are necessary to extract and transform tabular data into networks or graphs? Based on the scope defined in Chapter 2, we outline our assumptions and approaches in designing this framework. We provide a set of operators that systematically specify the computation of nodes and edges from a single data table or multiple linked data tables. We illustrate the theoretical definitions with examples of datasets presented in Chapter 2.

In Chapter 5, we tackle the second research question highlighted in this introduction: given a set of formal operations, how do we design a system that incorporates all these operations, enables intuitive access to these operations, and tightly couples network modeling with exploratory analysis? We present the system Ploceus for network-based visual analysis of tabular data. We discuss the overall system architecture design, and then focus on interface design issues. This chapter relates to Chapter 4 tightly and presents design decisions of turning the formal framework into a user-centric data exploration system. We also demonstrate how Ploceus is useful through an exemplary data analysis scenario. Finally, we report our findings on the usability and utility of Ploceus from an informal user study. We focus on the effectiveness of Ploceus in supporting end-user creation of network visualizations, and identify potential roadblocks in network modeling.

Finally we give a review and a summary of our research in Chapter 6. We also reflect on the important issues that we came across in designing both the formal framework and the system Ploceus. Several issues stand out as potentially fruitful avenues for further investigation: helping average users articulate and interpret relational joins of multiple tables, integrating customizable analytic modules in a plug-in based architecture, and visually analyzing big/dense graphs. We also discuss future

plans for evaluating Ploceus and propose new research directions on natural language interfaces for visual analysis based on our findings.

# CHAPTER II

# TABULAR DATA, NETWORKS AND ANALYTICAL GAP

In this chapter, we review the different forms of tabular data and various types of networks that are potentially relevant to this dissertation. We will then identify the specific forms of tabular data (spreadsheets and databases) we want to address, and the specific types of networks (weighted simple graphs) to be constructed. We also discuss in greater detail the motivation of this research by examining the nature of tasks in visual analysis and model-based reasoning.

## *2.1 Forms of Tabular Data*

Tabular data come in many forms, each unique in its schematic and semantic structure depending on the technology used and the data owner's goal. The term "tabular data" is thus fairly broad and can be interpreted as either *multivariate data* or *attribute relationship graphs*. We give examples of different types of tabular data in this section, define the scope of this research, and will base our discussion on these examples throughout the rest of this dissertation.

Single tables are pervasive in the form of spreadsheets and comma-separated value (csv) files. For example, Table 1 shows visits to the White House. For each visit, it records the last and first name of the person arranging the visit (LastNm, FirstNm), the type of visit (Type), the date (Date) and location (Loc) of visit, the size of the visiting group (Size), and the visitee's name (Visitee). Such tabular data are essentially *multivariate data* where rows represent entities or facts and columns represent entity attributes or other entities. In multivariate data, explicit network semantics is typically absent.

**Table 1:** A table of sample visitor information to the White House

| ID | LastNm | FirstNm | Type | Date | Size | Visitee | Loc |
|---|---|---|---|---|---|---|---|
| 1 | Dodd | Chris | VA | 6/25/09 | 2018 | POTUS | WH |
| 2 | Smith | John | VA | 6/26/09 | 237 | Office Visitors | WH |
| 3 | Smith | John | AL | 6/26/09 | 144 | Amanda Kepko | OEOB |
| 4 | Hirani | Amyn | VA | 6/30/09 | 184 | Office Visitors | WH |
| 5 | Keehan | Carol | VA | 6/30/09 | 8 | Kristin Sheehy | WH |
| 6 | Keehan | Carol | VA | 7/8/09 | 26 | Daniella Leger | OEOB |

**Table 2:** Two tables describing employees and the departments they work for in a company

(a) $R_{\mathsf{E}}$ : EMPLOYEE

| ID | FName | LName | Bdate | Dpt |
|---|---|---|---|---|
| 1 | John | Smith | 1965-01-10 | 2 |
| 2 | Franklin | Wong | 1952-04-09 | 3 |
| 3 | Jennifer | Wallace | 1970-10-23 | 3 |
| 4 | Ahmad | Jabbar | 1945-11-02 | 1 |

(b) $R_{\mathsf{D}}$ : DEPARTMENT

| ID | Name | City | State | Latitude | Longitude |
|---|---|---|---|---|---|
| 1 | Headquarters | Los Angeles | CA | 34.05 | -118.24 |
| 2 | Administration | San Jose | CA | 37.34 | -121.89 |
| 3 | Research | Houston | TX | 29.76 | -95.36 |

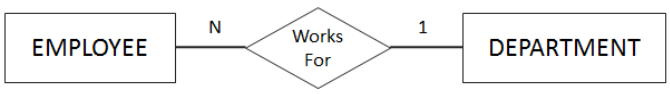



**Figure 3:** ER Diagram for the Employee-Department Relationship

Multiple linked tables are natural in most application domains for a relational database, although the same tables can also be described in spreadsheets. In a relational database, the ER Model (Entity-Relationship Model [25]) may be used to describe the underlying database design. Each row in a table represents a fact that corresponds to a real-world entity or relationship. For example, Table 2(a) represents facts about employees in a company, and Table 2(b) represents facts about departments in the same company. The two tables are linked by the *many-to-one* $\mathsf{EMPLOYEE-WORKS-FOR-DEPARTMENT}$ relationship type, as shown by the ER schema in Figure 3. That is, one department can have multiple employees, but one employee can only work for one department. *One-to-many* relationships are typically captured by foreign keys in a relational database [34]. In this case, $\mathsf{Dpt}$ in the $\mathsf{EMPLOYEE}$ table is a foreign key, referencing the $\mathsf{DEPARTMENT}$ table.

Another type of ER relationship is the *many-to-many* relationship, and it is captured by a separate relationship table [34]. For example, Table 3(a) represents selected facts about research grants awarded by the National Science Foundation (NSF) in the Information & Intelligent Systems division, and Table 3(b) represents facts about researchers. The two tables are linked by Table 3(c), which represents a many-to-many "PI-OF" relationship. That is, one researcher can receive multiple grants, and one grant can involve multiple researchers too. Figure 4 shows the entity-relationship (ER) data model from which the relational schema is derived.

**Table 3:** Tables describing researchers and the grants they receive

(a) $R_{\mathsf{G}}$ : GRANT

| GID | Title | Program | Program Manager | Amount |
|---|---|---|---|---|
| 1 | Data Mining of Digital Behavior | Statistics | Sylvia Spengler | 2241750 |
| 2 | Real-time Capture, Management and Reconstruction of Spatio-Temporal Events | Information Technology Research | Maria Zemankova | 430000 |
| 3 | Statistical Data Mining of Time-Dependent Data with Applications in Geoscience and Biology | ITR for National Priorities | Sylvia Spengler | 566644 |

(b) $R_{\mathsf{P}}$ : PERSON

| PID | Name | Org |
|---|---|---|
| 1 | Padhraic Smyth | University of California Irvine |
| 2 | Sharad Mehrotra | University of California Irvine |

(c) $R_{\mathsf{W}}$ : PI – OF

| Person | Grant | Role |
|---|---|---|
| 1 | 1 | PI |
| 2 | 1 | CoPI |
| 2 | 2 | PI |
| 1 | 3 | PI |

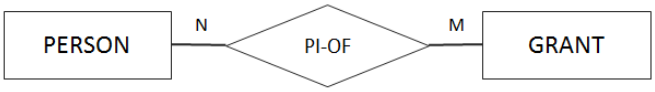



**Figure 4:** Grant-Person Relationship Model

The above mentioned entity relationships are *binary*, i.e. the relationships are defined between two classes of entities. A reflexive or recursive relationship is *unary*, defining references within one class of entities. For example, Table 4 shows records of email correspondences between individuals working in a company. Figure 5 shows the ER schema for this dataset.

**Table 4:** Two tables describing email communications between individuals

(a) $R_{\mathsf{P}}$ : PERSON

| ID | Name | Email Address | Department |
|---|---|---|---|
| 1 | John Smith | jsmith@abc.com | Sales |
| 2 | Franklin Wong | fwong@abc.com | Marketing |
| 3 | Jenny Wallace | jwallace@abc.com | Engineering |
| 4 | Ahmad Jabbar | ajabbar@abc.com | Research |
| 5 | Joseph Suzuki | suzuki@abc.com | Marketing |

(b) $R_{\mathsf{E}}$ : EMAIL_Record

| From | To | Timestamp |
|---|---|---|
| 1 | 4 | 03/12/11 09:23 |
| 2 | 1 | 09/21/10 15:45 |
| 3 | 2 | 12/03/10 11:03 |
| 4 | 1 | 11/23/10 14:03 |
| 5 | 2 | 02/16/11 16:01 |

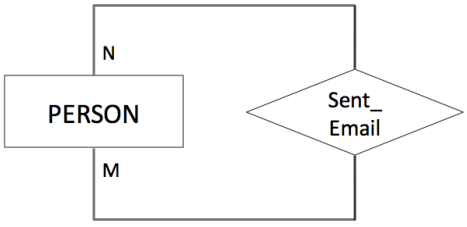



**Figure 5:** ER Diagram for the Unary Reflexive Relationship

These tabular data in multiple linked tables are essentially *attribute relationship graphs* [33] with explicit network semantics. Table 2 describes connections between employee and department entities. Similarly, Table 3 is a graph specifying the connection between two types of entities, researcher and grant, each with its own attributes. Table 4 is a graph specifying connections between the same type of entities in the form of email correspondences.

An OLAP (Online Analytical Processing) database [4, 23], unlike spreadsheets and relational databases, is not built for low-level atomic operations such as insertion and update but for analytical purposes. In OLAP databases, the two most common data schemas are the star schema and the snowflake schema. Figure 6 shows a star schema of an OLAP database for a hypothetical coffee chain [87]. The schema contains a core relation, or a fact table, that describes *measures*, which are attributes of analytical interest. The fact table is linked to dimension tables describing *dimensions*,

which are attributes for summarizing measures in different ways. It makes sense to think of dimensions as independent variables and measures as dependent variables. When dimensions have a hierarchical structure within themselves, for example, we can aggregate time at different levels of granularity (day, week, month, year), we model these dimensions using multiple tables and this gives us a snowflake schema (Figure 7).

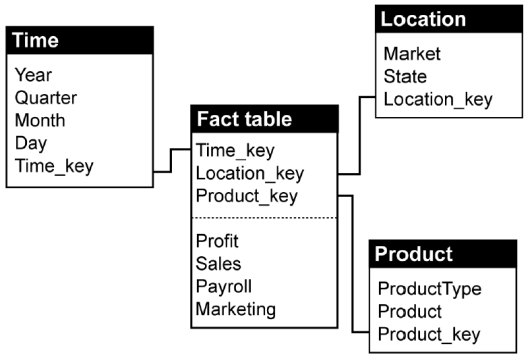



**Figure 6:** A star schema of an OLAP database taken from [87]. The fact table contains measures such as profit and sales; the measures can be characterized by different dimensions such as product and location, each with its own dimension table.

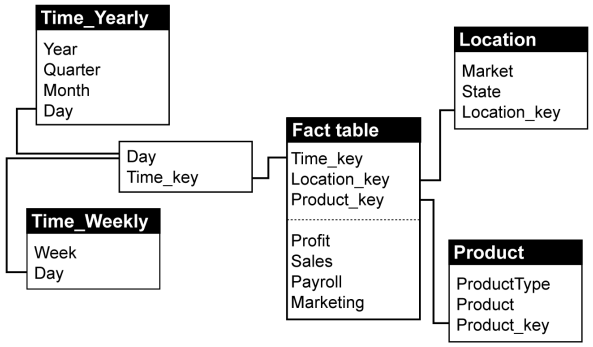



**Figure 7:** A snowflake schema of the same OLAP database as Figure 6, taken from [87]. The time dimension has a hierarchical structure, and is described by multiple relations. Analysts can thus summarize measures by week or by month.

The star schema and the snowflake schema provide a basis to construct data cubes [44] for better performance in analytical operations such as slice/dice and roll-up/drill-down. Analysts, for example, can quickly summarize the profit figure at a certain location for a specific period of time, or the same profit broken down by quarters

and products. The analytical power of OLAP, however, is not necessarily suitable for network-based analysis. Since the primary analytical interest is on measures, it is difficult to flexibly establish user-defined relationships between dimensions. We thus determine that the OLAP framework is not directly relevant for our purpose, and in this dissertation we focus on spreadsheets and databases which provide a basis for an alternative network-centric framework.

## *2.2 Networks: Different Types of Graphs*

In this dissertation we use the terms "network" and "graph" interchangeably. A graph is a mathematical and conceptual structure of a set of nodes (also called vertices), some of which are linked by edges.

There are many types of graphs, categorized by properties of nodes and edges. In a *weighted* graph, a numerical weight is assigned to each edge. The weight of the edge may reflect, for example, the frequency of contact between two persons. When the semantics of the edges are symmetric (i.e. if $a$ is related to $b$ then $b$ is related to $a$), the graph is *undirected*; otherwise the graph is *directed*. In some graphs, an edge can start on one node and ends on the same node, which is called a *loop*. When multiple edges connect the same two nodes, the graph is a *multigraph*.

In this dissertation, our primary focus is on the construction and transformation of *weighted simple graphs*, which are weighted undirected graphs that have no loops and contain at most one edge between any two nodes. This does not mean that multigraphs and directed graphs are beyond reach. As we demonstrate in Section 4.4.2, our formal framework supports the construction of edges with different semantics between the same nodes. Multiple weighted simple graphs can therefore be combined to produce multigraphs. Since few visual and statistical methods exist for analyzing multigraphs and we want to integrate network modeling tightly with visual exploratory analysis, we choose to design and implement our interface so that

multiple edge sets are created in separate graphs rather than a single graph (Section 5.5). Section 4.1 elaborates our rationale behind focusing on undirected graphs. As we shall see in Section 4.4.3, our framework can be extended to support the creation of directed graphs from data tables with reflexive relationships.

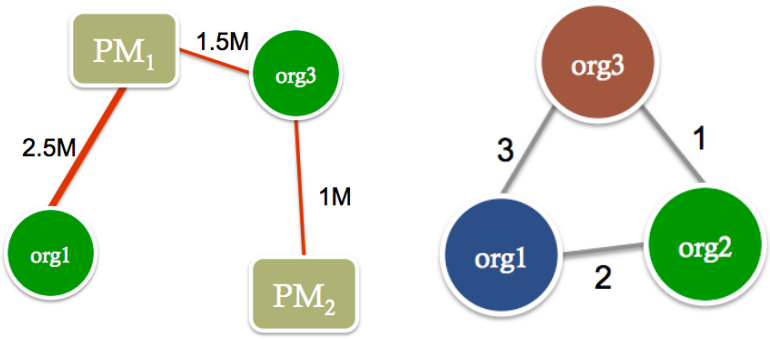
## *2.3 Analytical Gap and Semantic Distance*

For visualization designers and analysts, spreadsheets and databases naturally become the infrastructure upon which higher level visual analysis is accomplished. As discussed in the previous section, multivariate data in the form of single tables do not contain explicit network semantics; even when multiple tables are used to describe a graph, analysts' own notions of a meaningful network may render different graph structures. First of all, the concept of an entity is often multi-level nested. An attribute of an entity may be treated as an entity in its own right. For example, in Table 3(a), each row represents a grant entity with its own attributes such as title and program manager. A program manager can be in turn treated as an entity. In fact, it is often difficult to determine whether something is an entity or an attribute in data schema design [16]. Secondly, the same two entities can be connected via different semantics. In Table 1 for example, two people can be connected if they visited the same location, have the same last name, or started their visits on the same day.

The multivariate nature of tabular datasets thus implies opportunities for asking interesting questions that can be answered with network visualizations, and it is worthwhile to examine the nature of such questions more closely. Given the dataset in Table 3 for example, a grant applicant may want to understand the hidden dynamics, if any, in the process of awarding grants to choose an appropriate application strategy. NSF officials will want to understand the impact of the IIS program on the awardee social networks and on the creation and diffusion of intellectual property to evaluate funding policy. Many questions can thus be asked, for instance:

- **[Q1]** *Is there a strong affiliation between program managers and research institutions? i.e. Do certain program managers tend to give awards to a few selected institutions only?*
- **[Q2]** *From which organizations do researchers tend to have more cross-institution collaborations?*

One possible way to answer Q1 is to construct a network visualization (Figure 8(a)) where an organization and a program manager are linked if the manager has awarded at least one grant to researchers in that organization. We can define the edge weight, to be the total grant amount as shown in the figure, or to be the number of grants awarded. Analysts can provide initial answers to Q1 by inspecting the overall connectivity of the network. If the network consists of multiple small subnetworks that are disconnected from each other, there is evidence that a strong affiliation does exist. It is also likely that there is no disconnection within the network, but certain organizations or managers occupy more central roles. Statistical measures will enhance visual inspection to provide a more precise assessment.




(a) (b)

**Figure 8:** Visual models for answering questions on the NSF data set

Similarly, to answer Q2, we can create a network visualization where two organizations are connected by an edge if there is at least one collaboration between any

researchers from these two organizations. Figure 8(b) shows this network semantics, where the edge weight is based on the frequency of collaboration. Applying an appropriate layout algorithm to this network visualization and using statistical measures such as betweenness centrality will likely reveal important organizations that are "gatekeepers" connecting different subgraphs.

These questions are examples of *high-level* analysis tasks [14]. These high-level tasks have two major characteristics. First, they cannot be answered satisfactorily by simple "yes" or "no" or some precise values and metrics. Analysts can define measures to quantify "affiliation strength", for example in the case of Q1, but such numbers are only meaningful at the level of specific manager-institution pairs. Network visualizations are useful to show global structures in the network. Secondly, these high-level tasks are semantically rich and context dependent, and cannot be described abstractly or captured *a priori* because they usually only emerge during the process of exploration.

These high level tasks can be compared with *low-level* tasks [62, 77], which are usually topology-based or attribute-based. Topology-based tasks include finding neighbors, counting degree, finding shortest paths and identifying clusters; attribute-based tasks include finding nodes with specific attributes, or finding nodes connected by particular type of edges. Many of these low-level tasks are well defined questions with clear-cut answers, and they can often be effectively answered using search or database queries without much visual representation.

As Amar and Stasko argue, supporting only low-level tasks creates analytic gaps in addressing real analytic and sense-making goals [14]. To effectively support high-level tasks, we still need an intermediate level of description that explains what exact roles a visualization plays in high-level tasks, and how high-level tasks are carried out in relation to the low-level tasks. Although not explicit in Amar and Stasko's argument, a plug-and-play "causal model" and the flexibility in picking the variables involved

in the desired model seem to be important. This idea is elucidated in the model-based reasoning approach [65], which argues that using visualization in data analysis is best understood as the active construction and simulation of a model. Figure 8(a) and 8(b) are illustrations of analysts' desired model based on their analytical questions. Model-based reasoning recognizes human intentionality as a fundamental premise in visual analysis, and acknowledges potential discrepancies between what a visualization presents and what an analyst really wants to know. An apparent implication is that supporting flexible externalization of analysts' mental models is a central design concern. The question of "what information does this visualization show me?" is less important than and should be subsumed under the question of "how effectively does this system allow me to ask my questions about the data?".

To effectively support model-based reasoning, analysts thus must be able to quickly choose the relevant entities and relationships for model construction. The model will be subject to constant refinement and revision, where new variables and relationships are introduced and old ones transformed or discarded. Dynamic articulation of fluid network semantics is thus necessary, and the multivariate nature of many tabular datasets provides a fertile playground for performing this kind of model-based reasoning.

The focus of this research is thus not on representation and interaction techniques for visually analyzing a given network - a number of commercial and research systems have been designed for this purpose, as we shall see in the literature review in the next chapter. Rather, we aim to address flexible and rapid construction and manipulation of networks from tabular data, and to integrate network modeling with visual exploratory analysis.

# CHAPTER III

# RELATED WORK

This research addresses issues at the intersection of network analysis, visual analysis of multivariate data, visual databases, formal approaches to visualization specification and data transformation. Existing work in these areas is substantial. In this chapter we give an in-depth review of the work organized by themes, and discuss how our research is related and different from the existing literature.

## *3.1 Tabular Data Analysis*

Research on methods to analyze tabular data generally treats such data as multivariate or multidimensional data. The dimensions or attributes can be classified as ordinal, quantitative or nominal. It is a common practice to treat ordinal and nominal dimensions as dependent variables and to treat quantitative dimensions as independent variables.

### 3.1.1 Visual Methods for Analyzing Tables

Line charts and bar charts are commonly used representations to visualize the overall pattern of values belonging to a single dimension. Scatter plots effectively show the correlational distribution of values between two dimensions. Parallel coordinates [52] and parallel sets [17] visualize the relationships between two or more dimensions, which can be quantitative or categorical. These visualizations can be further enhanced with visual variables such as color, shape and size to depict more data dimensions. Keim et al. present techniques to visualize multidimensional data using dense pixel displays [56, 57].

Many of these multidimensional visualizations have been used for hundreds of

years, predating the modern era. The long-standing popularity is an indication of the effectiveness of these techniques, and contriving novel visual representations with similar levels of effectiveness seems to be difficult. Contemporary research focuses more on system research that integrates existing visualizations, provides theoretical frameworks that unify these visualizations, and augments visualizations with interaction techniques.

Many systems such as Visage [30], DEVise [66] XmdvTool [92] provide all these different chart types so that users can choose the appropriate visual encodings depending on the dimensions they want to analyze and the analytical questions. Table Lens [76, 75] (Figure 9) shows that simple graphics such as bar charts can compress large tables, using up much less screen space and affording pattern discovery.

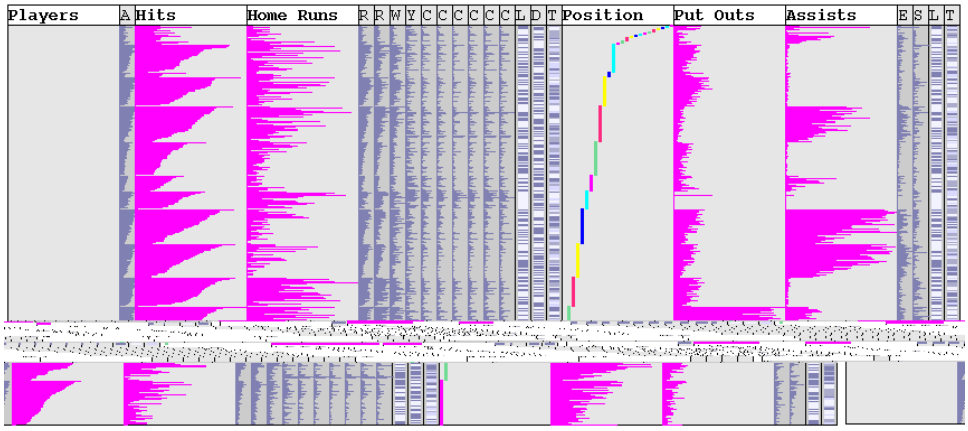



**Figure 9:** The table lens with multiple focal regions and a sorted column revealing correlations.

FOCUS [81] and InfoZoom [80] aggregates categorical or quantitative values by frequency using a space-filling visual technique (Figure 10). In my own past research, we have also explored the coordination of a space-filling view with a timeline bar chart representation to analyze online flight purchase transactions for anomaly detection [64].

Taking visualization generation to a new level, the APT framework [68] builds on top of the Semiology of Graphics [19] to provide a formal specification of various

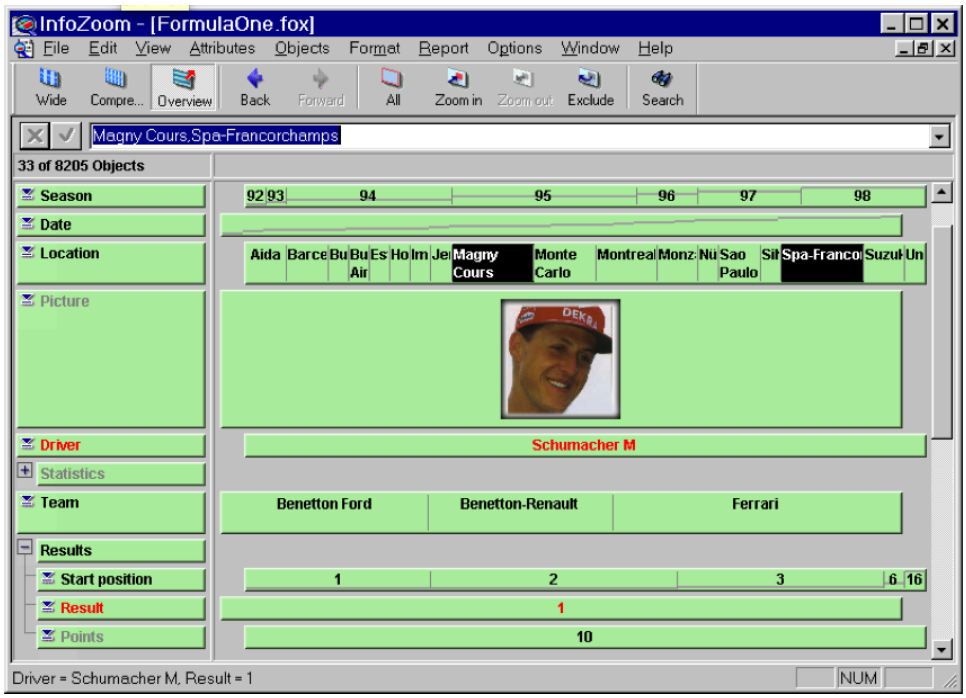



**Figure 10:** The InfoZoom interface used to analyze Formula One racing results, showing the victories of Michael Schumacher.

visualizations. The Grammar of Graphics [98] offers a similar approach, developing a comprehensive language to describe these visualizations.

None of these approaches supports imposing user-defined relationships between dimensions in the form of networks. In particular, nominal dimensions have generally received little attention in these approaches. Network seem to be an ideal visual representation for nominal values that refer to people and organizations, and it can be used to visualize relationships between quantitative and ordinal values as well.

### 3.1.2 Computational/Statistical Methods for Analyzing Tables

In computational/statistical methods to analyze tabular data, it is a common practice to treat quantitative dimensions as dependent variables and non-quantitative dimensions as independent variables. We can thus cross tabulate quantitative data using different dimension values. This basic idea underlies both the concepts of *pivot table* [54] and *contingency table* [21, 26].

**Table 5:** A sample data table containing personal information

| Name | Gender | Education | Annual Income |
|---|---|---|---|
| John Smith | M | High School | 30,120 |
| Mary Freeman | F | Post Graduate | 80,232 |
| Lucy Simon | F | College | 50,221 |
| Beth Wilson | F | College | 61,800 |
| Bill White | M | College | 59,010 |
| David Jackson | M | Post Graduate | 110,050 |

**Table 6:** A pivot table summarizing average income by gender and education level

| | High School | College | Post Graduate | Average |
|---|---|---|---|---|
| **Male** | 30120 | 59010 | 110050 | 66393 |
| **Female** | - | 56011 | 80232 | 64084 |
| **Average** | 30120 | 57010 | 95141 | 65239 |

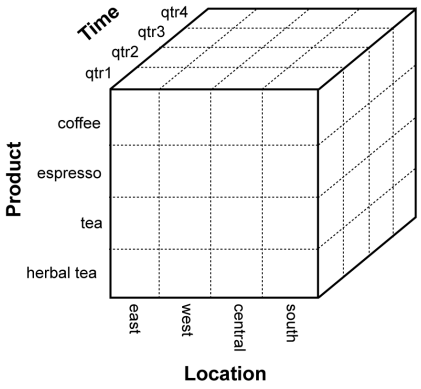



**Figure 11:** A datacube for the hypothetical coffee chain based on the star schema in Figure 6 [87]

**Table 7:** A contingency table summarizing count by gender and education level

| | **High-School** | **College** | **Post-Graduate** | **Total** |
|---|---|---|---|---|
| **Male** | 1 | 1 | 1 | 3 |
| **Female** | 0 | 2 | 1 | 3 |
| **Total** | 1 | 3 | 2 | 6 |

Pivot tables aggregate quantitative dimensions in a dataset using categorical dimensions. For example, Table 5 contains personal information about individuals. We can compute the average income by gender and by education level. This gives us Table 6. Pivot tables can be considered as a simplified version of the OLAP concept discussed in Section 2.1. OLAP pre-computes data cubes to optimize query performance so that the aggregated results can be retrieved directly (Figure 11).

In statistics, a contingency table focuses on the relations between two categorical variables as reflected in the frequency distribution (i.e. count data) instead of the aggregation of other quantitative variables. For example, if we produce a contingency table of gender by education level, the number in the cells will be the number of people (Table 7). Many sophisticated statistical techniques are developed for hypothesis-driven confirmatory analysis of contingency tables, under the umbrella of discrete multivariate analysis [21] and log-linear modeling [26].

Since contingency tables and pivot tables use the adjacency matrix representation, we can conceive of them as graphs. The nodes are the categorical values used for cross tabulation, and the values in the cells represents the weights of the edges connecting each pair of nodes. In fact, as we shall see in the next chapter, the count data or frequency distribution in contingency tables is similar to our notion of using co-occurrence count as edge weight. As noted in Section 2.1, however, the idea of cross tabulation is not adequate for network-based analysis. We may be able to take an OLAP database, issue queries, get the results as a contingency table and construct

basic networks from the results, but the way an OLAP database is built does not afford flexible transformative operations provided by our framework.

The focus of cross tabulation is usually to examine the relationship between different data table dimensions. The area of data mining also deals with analyzing multidimensional data at the data cases level [46]. For example, cluster analysis groups data cases with the goal to maximize inter-group differences and to minimize intra-group differences with techniques such as *k-means* [70]. Outlier analysis is interested in finding anomalous data cases.

### 3.1.3 Polaris and Tableau

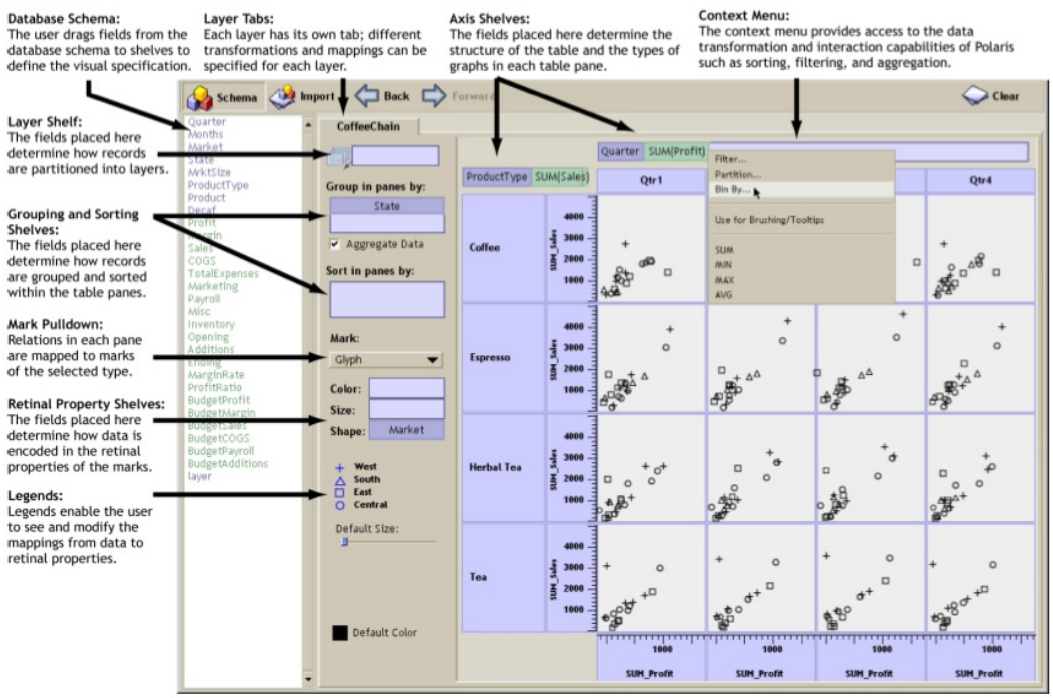



**Figure 12:** The Polaris interface when connected to a flat relational database. Analysts construct table-based displays of data by dragging fields from the database schema onto shelves throughout the display.

Polaris [84, 86, 85], later commercialized as Tableau [6], integrates visual and computational approaches for multidimensional data analysis. It builds on the OLAP concept and provides a visual formalism that specifies the construction of visualizations

of query results from multidimensional databases. It advocates the need for analysts to rapidly change what data they are viewing and how the data is visualized.

Polaris' approach is characterized by two distinct features. First, instead of separating query and visualization as disjointed stages, analysts can construct and execute queries by interactively dragging and dropping relational schema fields, and immediate visual feedback is provided. Secondly, in contrast to the mainstream approach which offers visualizations as monolithic objects, Polaris and Tableau are powered by algebraic visual specifications as a precise sequence of relational database operations. Every action taken by the analysts hence can be translated into a visualization state. By integrating data transformation and visual abstraction in a single process, the systems provide both flexibility and analytical power (Figure 12). While Polaris does not support network-based analysis, our motivation does resonate with its philosophy.

## *3.2 Network Data Analysis*

Much research effort has been focusing on how to analyze a given network and to discover insights from such data. Generally speaking the related work can be categorized into three types: statistical methods, visual methods, and hybrid methods.

### 3.2.1 Statistical/Computational Methods for Analyzing Networks

Social network analysis has a long and rich tradition. Many theoretical and methodological tools are available to analyze different aspects of a network [67, 53, 58, 95]. We certainly cannot provide a comprehensive review of all the existing work. Instead, we highlight some key concepts and methods used in statistical graph analysis.

#### *3.2.1.1 Structural Network Analysis*

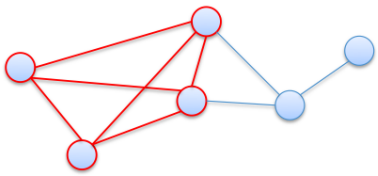
A primary focus of analysis has been on the *structural properties* of a network. We can analyze the structure of a network at multiple levels. At a global level, the measure of density looks at the extent to which nodes are connected to each other [58]. In

a undirected simple graph with the node set $V$ and the edge set $E$, the density is calculated as:

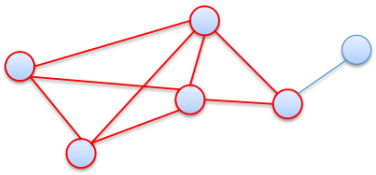
$$D = \frac{2|E|}{|V|(|V|-1)}$$

In the extreme case where every node in a graph is connected to every other node, the density of the graph reaches a maximum value of 1. The graph in this case is *complete*. When the density value is 0, no edge exists in the graph.

Drilling into parts of a graph, we are interested in "sub-structures" that may be present. An important concept here is *clique*. A clique is a maximal complete subgraph of three or more nodes where every node is directly connected to every other node. Cliques are indication of *cohesive subgroups*, which consist of members connected with dense and intimate relations. For example, the part of the graph highlighted in red in Figure 13(a) is a clique.



(a) A Clique structure highlighted in red



(b) A 2-Clique structure highlighted in red

**Figure 13:** Substructures in networks: Clique and N-clique.

The above definition of a clique is very strict because it enforces that every actor must have a direct tie with every other member of the clique. In actual network analysis practices, it makes sense to relax this definition to apply it to more general situations [53]. The concept of *N-clique* is thus introduced. An *N-clique* is a subgraph where every node is at most N relations away from every other node in the same group. For example, Figure 13(b) highlights a 2-clique in red. Other related concepts include *N-clan* and *K-plex*.

As we drill down further to the level of individual nodes, we analyze the structural properties of these nodes using measures such as *centrality*. Centrality measures quantify the prominence of actors in a network. The most widely used centrality measures are *degree*, *closeness* and *betweenness* [37]. The betweenness centrality, for example, indicates the extent to which an actor mediates the relations between subnetworks that are not directly connected. The betweenness centrality of node $i$ is computed as follows:

$$C_B(N_i) = \sum_{j<k} \frac{g_{jk}(N_i)}{g_{jk}}$$

where $g_{jk}$ is the number of shortest paths from node $j$ to node $k$, and $g_{jk}(N_i)$ is the number of shortest paths from node $j$ to node $k$ that pass through node $i$.

#### *3.2.1.2 Graph Mining*

Compared to structural analysis, graph mining is relatively new and evolved recently. Traditional data mining focuses on multivariate data that are assumed to be independent from each other. With the recognition that in many domains, relations exist between multivariate data cases in the form of graphs, graph mining begin to grow as a research area [40, 100]. Graph mining can be descriptive or predictive. One descriptive graph mining task is *group detection*. This task is similar to the technique of clustering in traditional data mining, but takes into account not only the attributes of nodes but also the relations between them. Another example of a descriptive task is to extract a small subgraph from a large graph that best captures the relationship between two nodes [35]. The work by Faloutsos et al. [36] addresses the predictive task of using power-laws of the Internet topology to estimate important parameters such as the average neighborhood size.

Inspired by the OLAP concept, recent work also began to examine performing OLAP-like operations on multivariate graph data [24, 89, 101]. In particular, the graph OLAP approach [24] defines two kinds of OLAP operations: information-based

and topology-based. While we aim to support performing multidimensional and multilevel graph analysis, our approach is different from performing OLAP on a static graph: instead of assuming a multivariate graph with fixed semantics, we address the construction and transformation of fluid network semantics from multivariate relational data.

### 3.2.2 Visual Methods for Analyzing Networks

All of the structural analytic techniques discussed in the previous section are performed computationally, but many of them (e.g. identifying cliques or individuals with high betweenness centrality measure) can be accomplished with a good visual representation.

In the conventional node-link representation of networks, common visual variables mapped to node or edge attributes are color, size and spatial positions. Toolkits such as prefuse [48] and many other commercial systems such as Gephi [1] provide support to apply user-defined visual encodings to a graph.

In particular, spatial position encoding, or graph layout, plays an important role in showing prominent visual structures such as clusters and outliers. A rich collection of graph layout and drawing techniques is available. In these graph layout techniques, it is important to show visual structures clearly so that we are not presented with a "giant ball of string". Dunne and Shneiderman propose a "NetViz Nirvana" as the guiding principles for measuring readabilities of graph visualization : 1) Every node is visible, 2) For every node you can count its degree, 3) For every edge you can follow it from source to destination, and 4) Clusters and outliers are identifiable [32].

Shneiderman and Aris categorize existing graph layout techniques into the following types: force-directed, map-based, radial or circular, matrix-based, cluster-based, and semantic substrates [77]. Some of the layout techniques are context independent. For example, the results of force-directed layouts will depend only on abstract graph

properties such as node degree and the connectedness of the graph. The meaning of node attributes and edge attributes has no influence on the visual layout. Approaches such as semantic substrates, on the other hand, incorporates more contextual cues into visual representations. In Figure 14, for example, the nodes represent judicial cases concerning the legal issue known as regulatory takings. The cases are categorized based on their levels (district court, circuit court and supreme court) and placed in three different regions. Within each region, the cases are placed according to their temporal attributes, where the oldest cases are on the left and the newest on the right. A vertical jittering function is applied to spread the cases out to reduce edge crossing. Patterns of citations across court levels can be easily perceived.

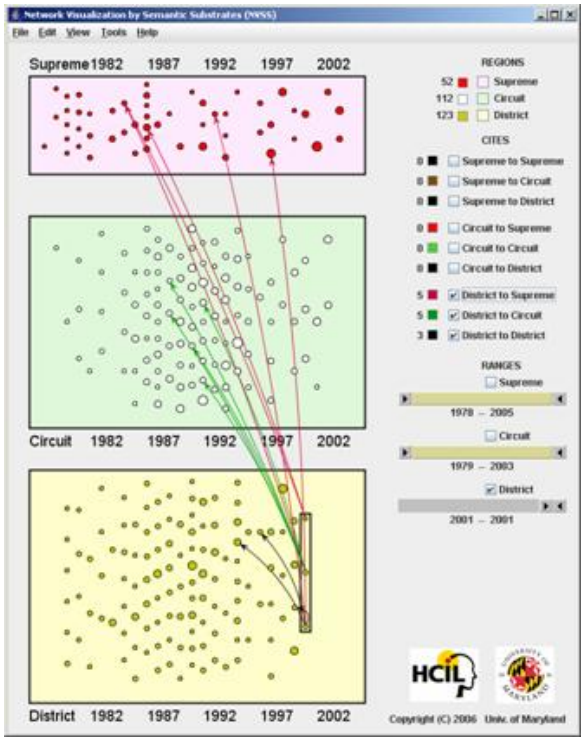



**Figure 14:** The semantic substrate approach to network visualization showing judicial cases positioned by their categories

When the size of a network is large, however, most of the layout techniques will turn out to be inadequate. Interaction plays a key role in navigating the large information space in the graph. The "Search, Show Context, Expand on Demand"

approach [91] provides a viable solution by assigning importance measures to graph nodes using a Degree Of Interest (DOI) function and interactively extracting contextual subgraphs for visualization.

Based on an understanding of the benefits and drawbacks of node-link and matrix representations, NodeTrix [50] explores how these different network representations can be integrated for the same underlying graph data. Node-link diagrams are appropriate for sparse graphs but result in poor readability for dense graphs. The adjacency matrix representation, on the other hand, handles dense graphs well but does not support path-related tasks effectively [41]. Nodetrix provides a hybrid of these two representations and supports smooth transition between these two kinds of representations depending on user tasks (Figure 15).

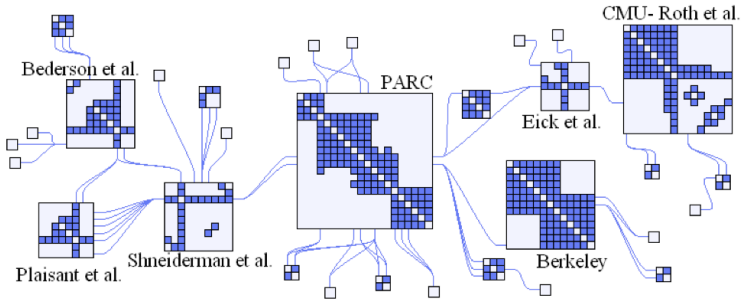



**Figure 15:** NodeTrix integrates node-link representation with matrix representation, showing part of the InfoVis Co-authorship Network

In these approaches, the focus is on visual representations of a static network structure, with support for low-level graph exploration tasks such as querying specific nodes or edges, finding neighbors and identifying clusters. The underlying graph data structure is hardly modified, and the support for higher-level analytics is not adequate.

PivotGraph [96] is an excellent example that illustrates the fluidity and interchangeability of the definitions of an entity and an attribute. It goes beyond simple encoding of node attributes in terms of color or size. Figure 16 shows a PivotGraph visualization of communications patterns between people in a large company. Instead

of representing people as nodes, attributes of people such as gender (x-axis) and location (y-axis) are coded as nodes. Visualizations as such provide an aggregated view of connection patterns based on attributes. Analysts thus can easily answer questions such as "How does the cross-gender communication pattern at location A differ from that at location E?", which will be more difficult to answer using the systems mentioned in the last section.

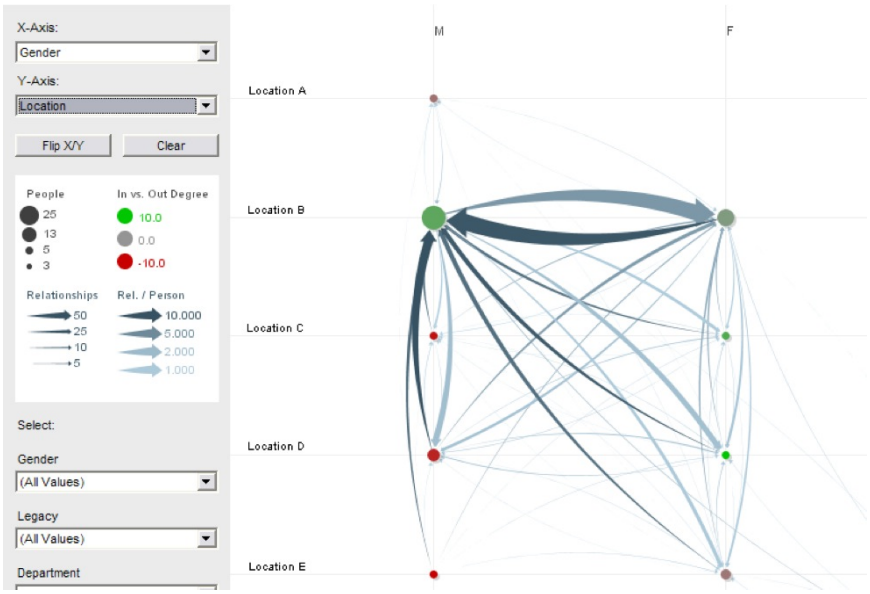
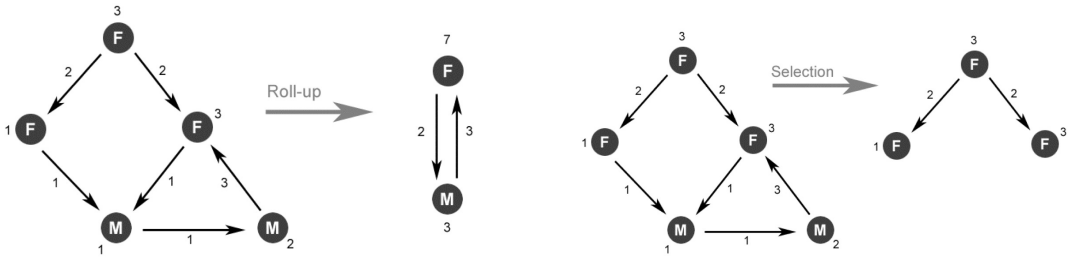



**Figure 16:** The PivotGraph Interface.




(a) Roll-up (b) Selection

**Figure 17:** Roll-up and selection operations in PivotGraph.

PivotGraph leverages the fact that nodes in many graph datasets are multivariate. By providing attribute-based transformation of the underlying graph, it recognizes practical analytical needs and goes beyond representational primacy. Two key data

transforming operations supported in PivotGraph are *roll-up* and *selection*. A roll-up operation, inspired by the OLAP paradigm, aggregates nodes having the same attribute values and aggregate edges accordingly. For example, Figure 17(a) shows an example where a simple social network is rolled-up by the gender attribute. A selection operation extracts a subset of graph items by finding nodes with specified attribute values. For example, in Figure 17(b) a sub-graph is selected based on the female gender.

PivotGraph's assumption of a given input graph with static edge definitions has limitations however. The pervasiveness of multivariate tabular data and the potentiality of doing network analysis on such data are overlooked. Analytical power of the system hence is undermined. The operations of roll-up and selection may not be sufficient when we consider the need to articular more fluid edge semantics and hence more dynamic graph structures. Analysts have no easy way to answer questions involving multiple edge types or attribute-based edge transformation even if they have the relevant data available.

### 3.2.3 Hybrid Approaches and Other Related Systems

From the reviews in the previous two section, it is clear that visual and statistical methods focus on different aspects of network analysis, and it would be advantageous to integrate them so that their strengths can complement each other.

Many systems integrate statistics with network visualizations to support systematic yet flexible analysis. Toolkits such as JUNG [73] allow users with programming skills to generate network visualizations and apply different layout algorithms, visual encodings and statistical metrics. Some other systems such as GUESS [13], Tulip [15], SocialAction [74], Pajek [5], UCINET [7], NetMiner [3] and Gephi [1] are in the form of readily executable software and users analyze networks through standard GUI controls.

In SocialAction (Figure 18) for example, emphasis is placed on structural attributes of nodes instead of their individual attributes. The structural attributes are derived using statistical metrics to reflect positions of nodes in a network. Analysts can choose a particular structural attribute such as betweenness centrality or cut-point and rank all the nodes. In this way, they can quickly get to important nodes that take up special roles.

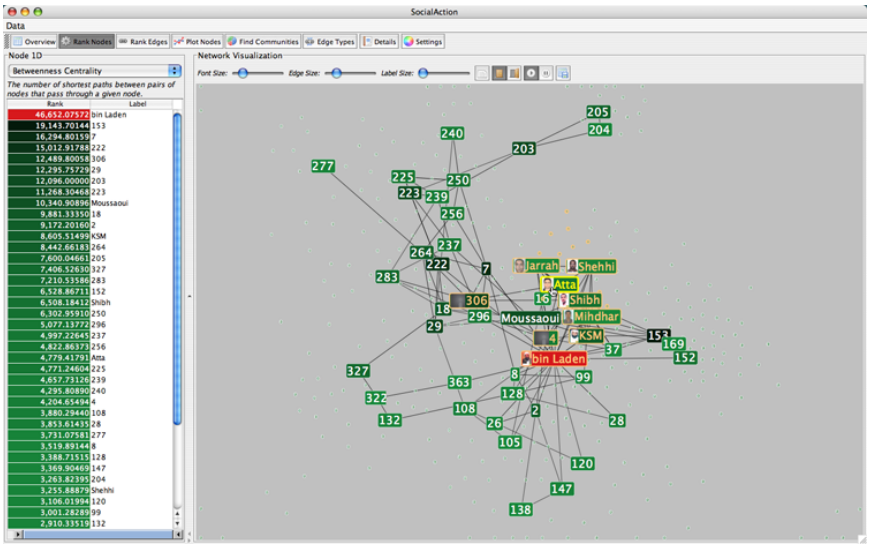



**Figure 18:** The SocialAction interface.

Depending on design and implementation, different selected data transformation features are supported in different systems. In SocialAction, groups of nodes can be collapsed into single nodes with meta-edges. These groups can be identified, for example, using automated community identification algorithms. Multiple link types are also considered where analysts can iterate between subnetworks of different link types. Gephi has similar functionalities of grouping nodes by both structural attributes computed statistically and inherent attributes defined in the data. In general, however, assuming a given graph data, these systems do not provide systematic ways of exploring the rich semantic space implied by multivariate data.

NodeXL [47] recognizes the disjunction between data manipulation and network visualization, and is designed to enable users to easily import, transform, visualize

and analyze network data. Built as a plug-in for Microsoft Excel$^{TM}$, NodeXL extends the visualizations included in Excel and adds "network" as a chart type. Figure 19 shows the Excel-based interfaces for specifying network semantics. Nodes and edges are defined in separate sheets; within each sheet, each row denotes a node or edge, and columns denote properties of nodes or edges.

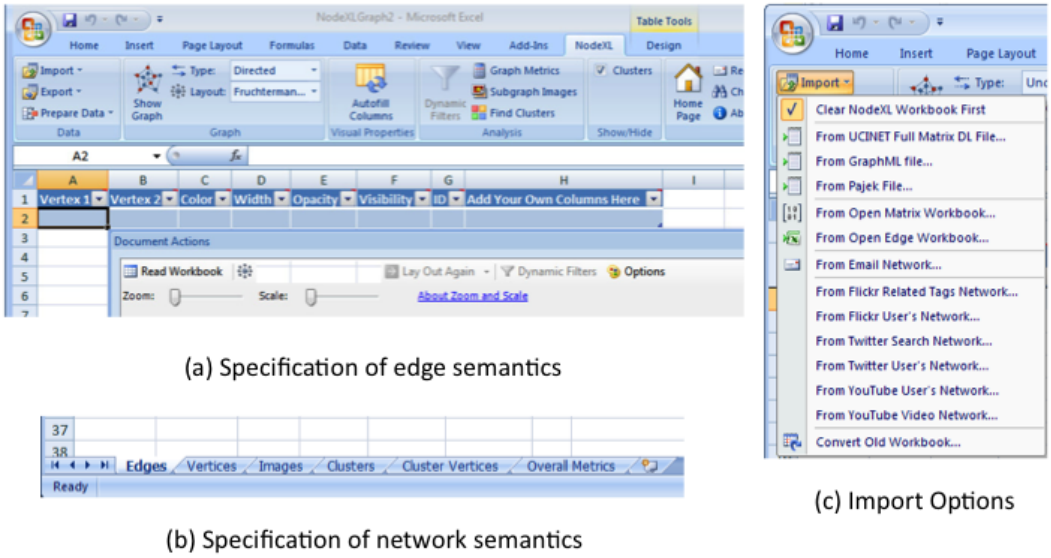



**Figure 19:** Interfaces for data manipulation in NodeXL

The editable Excel interface allows direct entering or changing individual datum. Analysts can also leverage data transformation features of Excel such as spreadsheet formulae and Pivot Table [54] to change the graph structure. Given a tabular dataset, potentially it is possible to accomplish a wide range of graph creation and transformation tasks using Excel, but analysts have to be Excel expert users. The process of semantic articulation is handled purely at the data layer with no immediate visual feedback, hence is separated from the visual exploration process. Furthermore, during the process of data transformation, many different networks may have been created, but there is no easy way to examine them in conjunction to achieve a more holistic understanding of the data.

ManyNets [38] is designed to visualize and explore multiple networks. Using a

table-based metaphor, ManyNets represents individual networks as rows and the columns are the summary information about each network such as node count or edge density (Figure 20). Analysts can drill down into individual networks and use SocialAction [74]. A prerequisite for ManyNets to work, however, is that these networks should share the same semantics. As discussed in the paper, such networks are usually subnetworks derived from a large network using a divide-and-conquer approach. The system does not support visualizing and analyzing networks with different semantics but shared nodes.

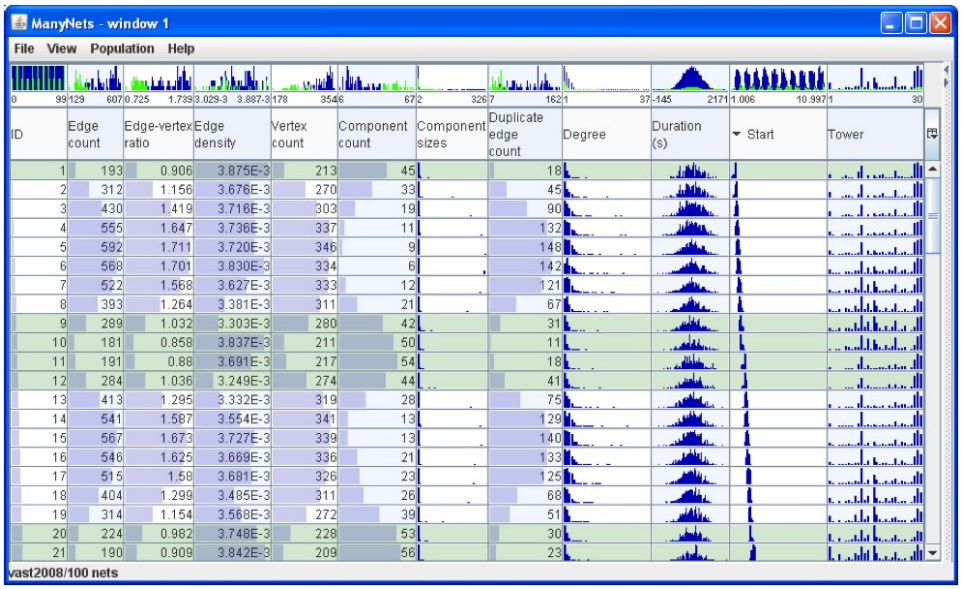



**Figure 20:** ManyNets displaying a time-sliced cell-phone call network.

## *3.3 Data Tables and Graphs*

Although database researchers do not generally consider entity-relationship models as graphs, the idea of extracting trees and networks from relational data has been explored in existing work. The Grammar of Graphics discusses an algebraic approach for mapping tables to directed trees [98]. The need for retrieving and publishing selected information on the web leads to work that models databases as virtual graphs [45] and provides XML document interface of relational data for web applications [22].

A comprehensive formal framework for flexible graph construction, however, has been absent.

A number of visual analytics system have also experimented with the idea of creating graphs from data tables. Jigsaw [82, 83, 42] assumes that entities are typed and can thus be categorized. The system provides powerful support for entity definition: users can let natural language processing packages extract entities of common types such as people, places and organizations from unstructured text; alternatively users can feed a manual definition of entities of any arbitrary type such as car makes and genes into the system. Instead of letting users define embedded relationships, Jigsaw builds the semantics of relationships into the system design based on a simple assumption: entities are identified purely lexically, and entities appearing in the same documents are connected.

This approach is generalizable beyond text documents to tabular data sources such as spreadsheets: content in each table cell can be either treated as an entity having a type specified by the column name, or it can serve as unprocessed data from which entities are extracted. Each row in a table can also be treated as an entity, analogous to the document concept in unstructured text. Furthermore, entities that appear in the same row in a spreadsheet are usually semantically related.

The entities/documents and their connections are stored in local databases, they can also be exported as XML-based data files. Since automated entity extraction is not always accurate, analysts can interactively add entities that are missed by the extraction process, change the type of an entity, delete a misidentified entity, or merge multiple lexical strings that refer to the same logical entity under one single alias. User interaction thus is limited to changing the nodes in a graph, not the semantics of relationships between nodes.

The co-occurrence based definition allows flexible explorations of entity relationship without having to explicitly specify the nodes and edges. Users can import

spreadsheets into Jigsaw and generate visualizations using the monolithic views provided by the system. One of the frequently used views, the List View (Figure 21), allows easy selection and population of entities by types (corresponding to spreadsheet columns). The connections between entities are automatically drawn based on the co-occurrence assumption.

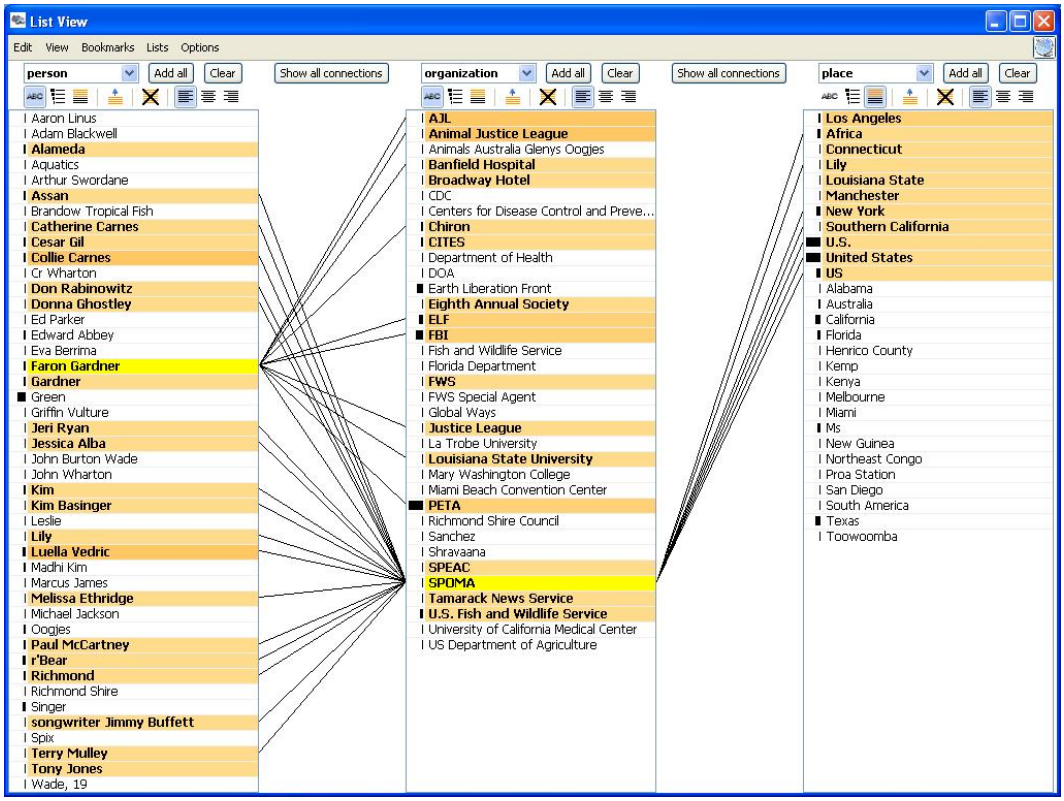



**Figure 21:** The List View in Jigsaw

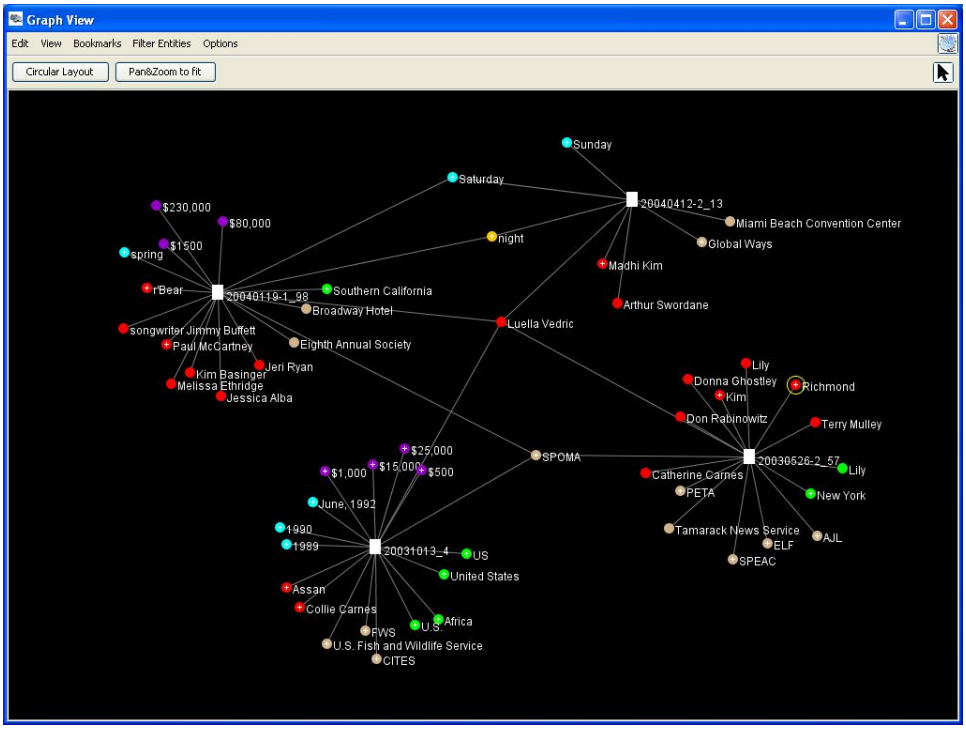



**Figure 22:** The Graph View in Jigsaw

The co-occurrence assumption however implies potential limitations on the system's analytical power. The generalized semantics of co-occurrence means that it is difficult to ask questions involving user-defined relationships. The Graph View (Figure 22) shows the connections between entities and documents/table rows in a node-link representation. Documents or table rows are represented as white rectangles and entities are represented as colored dots. The exploration of the network is incremental: starting from a single node, its immediate neighbors can be expanded and collapsed. Due to the constraints imposed by the co-occurrence assumption, when more entity nodes are added to the visualization, the nodes and the links merely indicate the presence of these entities. The relationships between the entities are indirect and only meaningful as defined by the occurrence assumption. As a result, applying layout algorithms is limited to showing co-occurrence patterns across selected documents or table rows. Analysts may want to construct and examine more

direct relationships between entities without any documents or table rows. There is also limited data transformation support due to Jigsaw's indiscrimination between nominal and quantitative attributes.

The Attribute Relationship Graph approach [97] incorporates direct connections between table columns into an existing cross-filtered visualization. The fundamental notion of this approach is very similar to our motivation. It does not, however, provide a detailed account of the possible types of graphs to be created or the mechanisms to handle multiple linked data tables. Our work goes beyond attribute relationships to offer a comprehensive construction and transformation framework, thus enhancing the analytical power of the system.

Two commercial systems, TouchGraph Navigator [12] and Centrifuge [8], are similar to our system Ploceus. ToughGraph Navigator lets users create and visualize networks from data tables, and Centrifuge includes a suite of visualizations including relationship graphs, charts and timelines. Orion [49], a research system developed by researchers from Stanford and IBM, also supports the goal of modeling networks from tabular data.

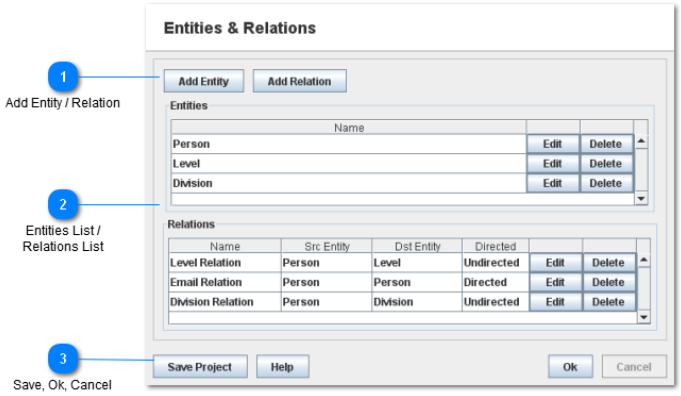



**Figure 23:** The Interface in TouchGraph Navigator to Create Networks from Relations

The design and functionalities of Ploceus are different from these systems in a variety of ways. First, while TouchGraph Navigator supports network modeling from

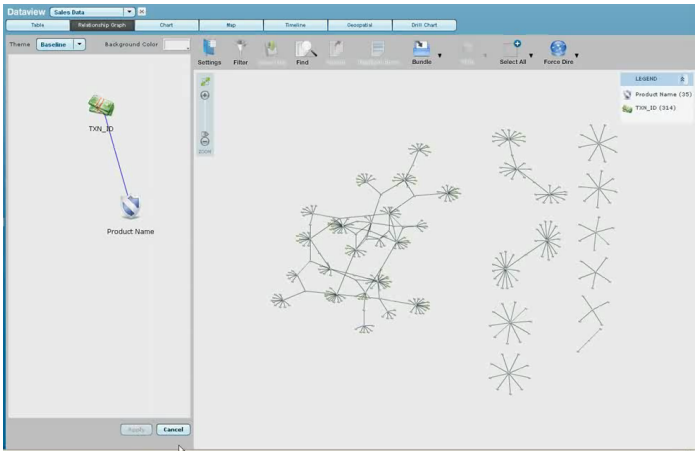



**Figure 24:** The Interface in Centrifuge Navigator to Create and Visualize Relationship Graphs

relational data tables, the modeling interface is dialog based (Figure 23). The interface of Ploceus follows the direct manipulation paradigm and provides better visual feedback. Centrifuge provides a direct manipulation interface (Figure 24) and thus comes close to the design of Ploceus in this regard. Secondly, based on our formal framework, the network modeling capabilities of Ploceus are richer than these systems.

In Table 8, we examine the operations provided by these systems and compare these operations with those provided in our formal framework. Due to the inaccessibility of the commercial software TouchGraph Navigator [12], we cannot do a comprehensive assessment of its features and thus omit it from the comparison. We focus on three other systems: Attribute Relationship Graph (ARG) [97], Centrifuge [8] and Orion [49].

**Table 8:** A comparison between different systems in terms of the network modeling operations provided

| | **Our Framework** | **Centrifuge** | **ARG** | **Orion** |
|---|---|---|---|---|
| Create Nodes | ✓ | ✓ | ✓ | ✓ |
| Add Attributes | ✓ | ✗ | ✗ | ✗ |
| Create Connections | ✓ | ✓ | ✓ | ✓ |
| Assign Weights | ✓ | ✗ | ✗ | ✓ |
| Project | ✓ | ✗ | ✗ | ? |
| Aggregate - pivoting | ✓ | ✗ | ✗ | ✗ |
| Aggregate - binning | ✓ | ✓ | ✗ | ✗ |
| Aggregate - proximity grouping | ✓ | ✗ | ✗ | ✗ |
| Slice 'n Dice | ✓ | ✓ | ✗ | ✓ |
| Filter by value | ? | ✗ | ✓ | ✓ |

As evident from Table 8, all the systems provide support for basic operations such as creating nodes and connections. Adding node attribute, pivoting and proximity grouping are absent in all systems except our framework and Ploceus. Due to different interface designs, sometimes there is no direct one-to-one mapping between the operations in our framework and those in other systems. For example, in Orion, there is no "project" operation, but users can create one-mode networks by "promoting" a column to a table and connect the column to itself [49]. Similarly, in our framework, there is no explicit "filter by value" operation, but analysts can identify specific node values through the search function provided at the interface level.

# CHAPTER IV

# FROM RELATIONS TO GRAPHS: A FORMAL FRAMEWORK

In this chapter we present a formal framework for constructing and transforming networks from data tables. In this framework, we use relations [28] to describe data tables, and focus on weighted simple graphs as the type of networks to be supported.

## *4.1 Approach and Assumption*

Although explicit network semantics are typically absent in relational data, the concept of a relation by definition implies a relationship among the attributes involved in the relation. In a relation, rows represent entities or facts and columns represent entity attributes or other entities. For example, we may wish to represent data about all the visits to the White House in the form of a table (Table 1). For each visit, we can record the last and first name of the person arranging the visit (LastNm, FirstNm), the type of visit (Type), the date of visit (Date), the size of the visiting group (Size), the name of the visitee (Visitee), and the visiting location (Loc). We organize data belonging to the same type or domain by grouping them into columns (e.g. values 2018, 237, 144, 184, 8, 26 are organized under the column Size), and we organize data that are semantically related as some real world fact into rows. The co-occurrence of data in the same row can be interpreted differently depending on context. When Dodd, Chris and WH appear in a single row of the visit logs, this co-occurrence can be interpreted as a visiting relationship between two entities: the person Chris Dodd visited the White House. When Cognitive and Social Design of Robotic Assistants and 3325900 appear in the same row of the NSF grant data, this co-occurrence can be interpreted

as a description of an entity in terms of an attribute: the amount of the grant with the given name is $3325900.

Our approach leverages this simple observation that *the meaning of row-based co-occurrence of data is context-sensitive*. It is thus possible to propose a co-occurrence based formal framework which specifies the construction and transformation of graphs, where the meaning of the graphs created will be subject to users' interpretation. Co-occurrence is typically undirected: when A co-occurs in a row with B, B co-occurs with A too. The meaning of connections derived from co-occurrence, however, can be interpreted as having a sense of direction. For example, when a person and a location is connected, the meaning of connection is a visit *by* the person *to* the location.

We base our formal framework on the relational model [28] used extensively in database theory with basic relational algebraic operators such as selection ($\sigma$), projection ($\pi$), join ($\bowtie$) and aggregation ($\mathscr{F}$) [34]. In our formalism, we make the following two assumptions which may be considered as an inherent part of the relational data model definition:

- Each row in a table has a unique identifier.
- Each value in the table cells is *atomic*, i.e. it is either a number, a date, or a string, and the value cannot be decomposed into meaningful smaller units.

## *4.2 First-order Graph Construction*

To construct graphs from tables, we need to map table elements into graph vertices or nodes, and connect nodes by edges based on co-occurrence. The simplest graphs we can construct are those in which each node and edge is constructed from one single row only. We call this kind of graphs *first-order graphs*. Before we proceed with the discussion of first-order graph construction, we first define some basic terminology:

**Definition 1.** [**Relation**] A *relation* $R$ is a set of *tuples* $T$, $T = \{t_1, t_2, \ldots, t_m\}$, with a list of dimensions $D$, $D = \langle d_1, d_2, \ldots, d_n \rangle$. Each dimension $d_i$ is associated with a data

type, called the domain of $d_i$ and denoted by $dom(d_i)$. Each tuple $t$ is an ordered list of dimension values $\langle v_1, v_2, \ldots, v_n \rangle$ where $t.d_i = v_i$ and $v_i$ has the data type of $dom(d_i)$.

Using Table 1 as an example, it is a relation with a list of dimensions $D = \langle$ID, LastNm, FirstNm, Type, Date, Size, Visitee, Loc$\rangle$. Each of the dimensions has its data type or domain, e.g. $dom$(Size) = Integer, and $dom$(LastNm) = String. Typically the column names are referred to as attributes. We call them dimensions here in order to differentiate them from attributes of nodes in the constructed graphs. Each row or a tuple is identified by a unique identifier. For example, the second tuple $t_2 = \langle$2, Smith, John, VA, 6/26/09, 237, Office Visitors, WH$\rangle$, and $t_2$.Date = 6/26/09.

**Definition 2.** [**Graph**] A *graph* $G = (N, E)$ consists of a node set $N$ and an edge set $E$. Each node or vertex $n \in N$ is a pair $(l, A)$ where $l$ is a string representation of the node's name; it is not meant to be a unique identifier. $A$ is a list of attributes describing the node: $A = \langle a_1, \ldots, a_k \rangle$. For each attribute there is a value associated with it: $n.a_i = v_i$. In a simple graph, each edge $e(n_1, n_2) \in E$ connects a pair of nodes, where $n_1$ is the source node and $n_2$ is the target node. An optional weight can be assigned to an edge, denoted by $w(n_1, n_2)$.

**Definition 3.** [**Dimension List**] A *dimension list* $\Delta$ is an ordered list of dimensions: $\Delta = \langle d_i, \ldots, d_k \rangle$ such that $\forall d_i \in \Delta, d_i \in D$. Given a dimension list $\Delta$, a *subtuple* $t.\Delta$ can be obtained where $t.\Delta = \langle v_i, \ldots, v_k \rangle$, which is a projection of tuple $t$.

Referring to Table 1, if we define $\Delta = \langle$LastNm, FirstNm$\rangle$, then $t_6.\Delta = \langle$Keehan, Carol$\rangle$. The concept of a dimension list is useful when we talk about constructing nodes based on selected dimensions.

### 4.2.1 Single Relation

We first consider the construction of graphs from a single relation. That is, all the data needed for graph construction are present in a single table. *For any given tuple*

*in a relation*, there are two main ways to construct a node from it. We can create a node such that its label is a subtuple: $n.l = t.\Delta$, or its label is a function of a subtuple: $n.l = f(t.\Delta)$. For example, with $t_1 = \langle 1,$ Dodd, Chris, VA, 6/25/09, 2018, POTUS, WH$\rangle$, if we are interested in the first and last names, we can define $\Delta = \langle$LastNm, FirstNm$\rangle$, the resulting node label will be the subtuple "Dodd, Chris". When the cardinality of $\Delta$ is 1 (e.g. $\Delta = \langle$Loc$\rangle$), we assign single dimension's values as labels (e.g. "WH").

A functional construction of node label is possible too. For example, we can define a function that takes Size as the argument, and returns "large group" as the node label if the group size is above 100, and "small group" otherwise.

In a similar way, we can assign an attribute to a node based on a subtuple or a function of subtuple: $n.a_i = t.\Delta^{'}$ or $n.a_i = f(t.\Delta^{'})$. For example, the node $n$ created from $t_1$ with "Dodd, Chris" as its label can have an attribute Type, and $n.$Type = VA.

We can thus define the notion of a *first-order node* formally:

**Definition 4.** [**First-order Node**] A node $n$ constructed from a single relation is first-order if and only if its label and attributes are constructed from a *single* tuple $t$:

$$\begin{cases} n.l = t.\Delta_l \text{ or } f(t.\Delta_l) \\ \forall a_i \in n.A, n.a_i = t.\Delta_i \text{ or } f(t.\Delta_i) \end{cases}$$

In short, if the construction of a node results from only a single row of the table, the node is a first-order node. All nodes constructed from *multiple* tuples are *higher-order nodes* (we discuss the construction of higher-order nodes in Section 4.3).

Usually we create a set of first-order nodes from a set of tuples in one pass. For example, we can construct a set of nodes $N$ representing the people visiting the White House from all the tuples in Table 1, and they have an attribute called Type: $\forall n \in N, n.l \in \{t.\langle$LastNm, FirstNm$\rangle\}$ & $n.$Type $\in \{t.\langle$Type$\rangle\}$. Table 9(a) shows the first-order nodes created. Similarly, we can construct a separate set of nodes $M$ representing the meeting locations: $\forall m \in M, m.l \in \{t.\langle$Loc$\rangle\}$. Table 9(b) shows the meeting location nodes. Since there may be rows containing identical values

for selected table dimensions, two first-order nodes can have the same labels and attributes. For example, we have four "WH" location nodes in Table 9(b).

**Table 9:** First-order nodes and edges created from a single relation in Table 1

(a) $N$: People Nodes

| Node Label | Type | locale |
|---|---|---|
| Dodd, Chris | VA | t1 |
| Smith, John | VA | t2 |
| Smith, John | AL | t3 |
| Hirani, Amyn | VA | t4 |
| Keehan, Carol | VA | t5 |
| Keehan, Carol | VA | t6 |

(b) $M$: Location Nodes

| Node Label | locale |
|---|---|
| WH | t1 |
| WH | t2 |
| OEOB | t3 |
| WH | t4 |
| WH | t5 |
| OEOB | t6 |

(c) $E(N, M)$: Edges

| n | m | basis |
|---|---|---|
| ("Dodd, Chris", VA) | WH | t1 |
| ("Smith, John", VA) | WH | t2 |
| ("Smith, John", AL) | OEOB | t3 |
| ("Hirani, Amyn", VA) | WH | t4 |
| ("Keehan, Carol", VA) | WH | t5 |
| ("Keehan, Carol", VA) | OEOB | t6 |

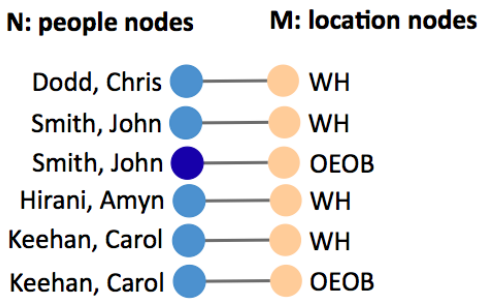



**Figure 25:** Visualization of the first-order graph in Table 9. The people nodes are colored by the attribute "Type", and the location nodes do not have attributes

**Definition 5.** [**Locale of a Node**] To ensure consistency and tractability when we deal with more complex graph manipulation later, we define the **locale** of a node to refer to the set of tuples from which the node is constructed. The cardinality of the locale of a first-order node is 1 because a first-order node is constructed from a single tuple only: $locale(n) = \{t\}$. For example, the locale of the node with label "Dodd, Chris" and attribute VA is $t_1$, as shown in Table 9(a). The cardinality of higher-order nodes can be greater than 1, as we shall see in Section 4.3.

As mentioned earlier, our formalism focuses on establishing relationships based on co-occurrence in rows. Two first-order nodes are thus connected if they share the same locale (i.e. appear in the same tuple). Table 9(c) shows the set of edges connecting people nodes and location nodes. Figure 25 visualizes this graph. Formally speaking,

**Definition 6.** [**First-order Graph from a Single Relation**] Given two first-order nodes $n_1$ and $n_2$ constructed from the same relation, a *first-order edge* $e(n_1, n_2)$ can be constructed if $n_1$ and $n_2$ share the same locale.

$$\text{If } \exists t,\ locale(n_1) = locale(n_2) = t, \text{ then } e(n_1, n_2).$$

**Definition 7.** [**Basis of an Edge**] We define the **basis** of an edge to refer to the set of tuples that the edge is constructed from. We use $basis(n_1, n_2)$ to denote the basis of the edge linking node $n_1$ and $n_2$. Table 9(c) shows examples of edge bases

(first two columns define the edge, and the last column shows its basis). The basis of higher-order edges can contain non-tuples (e.g. see Section 4.3.2).

### 4.2.2 Multiple Relations

We now consider the construction of graphs from a collection of relations. That is, the data needed to construct graphs are distributed across multiple tables. Compared to a single table, multiple tables in a relational database contain more explicit network semantics. We can consider Table 3(a) and 3(b) as the specification of two types of nodes in a graph, and the connections between the nodes are specified in 3(c). The network semantics that an analyst wants to explore, however, may not always be directly captured by the underlying ER Model. For example, we may be interested in the relationships between employees as reflected in the physical proximity of their working locations; or we want to explore the collaboration relationships between researchers, as manifested through the grants they have received together. Semantics such as these are not directly captured in the database design, hence it is necessary to construct graphs reflecting these semantics from the tables.

We consider two major ways by which multiple tables can be involved in first-order graph construction. First, we can create two sets of first-order nodes, each constructed from a single table only, and the edges between the nodes are created by linking two tables. In the second and more complex case, a set of first-order nodes can be constructed such that their labels come from one table, and their attributes come from another table. We do not allow constructing node labels from multiple relations in our formalism for the purpose of simplicity.

#### *4.2.2.1 Creating edges across multiple tables*

Given that we are only focusing on first-order graphs in this section, and assuming that we know how the tables are linked, either through foreign keys or separate relationship tables, the formalism discussed in Section 4.2.1 can be easily extended

to multiple tables. The notion of co-occurrence is no longer limited to one tuple in a relation, but is extended to include two or more tuples in multiple relations specified by an appropriate JOIN condition. For example, we may want to create a set of person nodes from Table 2(a) (duplicated as Table 10(a)): $n.l \in \{t.\ \langle \mathsf{FName}, \mathsf{LName} \rangle\}$ where $t \in R_{\mathsf{E}}$, and a set of location nodes from Table 2(a): $n.l \in \{t.\ \langle \mathsf{City}, \mathsf{State} \rangle\}$ where $t \in R_{\mathsf{D}}$. Table 11 (a) shows the person nodes created and Table 11 (b) shows the location nodes created.

**Table 10:** A Duplication of Table 2

(a) $R_{\mathsf{E}}$ : EMPLOYEE

| ID | FName | LName | Bdate | Dpt |
|---|---|---|---|---|
| 1 | John | Smith | 1965-01-10 | 2 |
| 2 | Franklin | Wong | 1952-04-09 | 3 |
| 3 | Jennifer | Wallace | 1970-10-23 | 3 |
| 4 | Ahmad | Jabbar | 1945-11-02 | 1 |

(b) $R_{\mathsf{D}}$ : DEPARTMENT

| ID | Name | City | State | Latitude | Longitude |
|---|---|---|---|---|---|
| 1 | Headquarters | Los Angeles | CA | 34.05 | -118.24 |
| 2 | Administration | San Jose | CA | 37.34 | -121.89 |
| 3 | Research | Houston | TX | 29.76 | -95.36 |

Since these two types of nodes are created from different tables, we specify an *equi-join* condition that links tuples from these two tables: $R_{\mathsf{E}}.\mathsf{Dpt} = R_{\mathsf{D}}.\mathsf{ID}$. A person node and a location node are connected if the union of the locales of these two nodes satisfy the join condition specified, and the basis of the edge is the union of the locales of these two nodes. Table 11 (c) shows the edges between person nodes and location nodes. We thus arrive at the following definition:

**Definition 8. [First-order Graph from Multiple Relations]** Given two first-order nodes $n_1$ and $n_2$ constructed from different relations, a *first-order edge* $e(n_1, n_2)$ can be constructed if the locales of $n_1$ and $n_2$ satisfy a predefined condition (denoted by $\theta$) joining these relations. Formally speaking,

Given $locale(n_1) = R_1.t_i$ and $locale(n_2) = R_2.t_j$

If $R_1.t_i + R_2.t_j \in R_1 \bowtie_\theta R_2$, then $e(n_1, n_2)$.

$basis(n_1, n_2) = locale(n_1) + locale(n_2) = R_1.t_i + R_2.t_j$.

In Definition 8, we use the operator + to denote concatenation of two tuples. Concatenation preserves the order of the values in the tuples and does not remove duplicates. For example, we have $R_{\mathsf{E}}.t1 = \langle 1,\ \mathsf{John},\ \mathsf{Smith},\ 1965-01-10 \rangle$ and $R_{\mathsf{D}}.t2 = \langle 2,$ $\mathsf{Administration},\ \mathsf{San\ Jose},\ \mathsf{CA},\ 37.34,\ -121.89 \rangle$, $R_{\mathsf{E}}.t1 + R_{\mathsf{D}}.t2$ will result in $\langle 1,\ \mathsf{John},\ \mathsf{Smith},$ $1965-01-10,\ 2,\ \mathsf{Administration},\ \mathsf{San\ Jose},\ \mathsf{CA},\ 37.34,\ -121.89 \rangle$. We use concatenation instead of other operators such as union ($\cup$) because according to a conventional definition of relational join, a join operation on two relations $R$ with dimensions $(A_1, A_2, ...A_n)$ and $S$ with dimensions $(B_1, B_2, ...B_m)$ results in a relation $Q$ with $n+m$ dimensions $(A_1, A_2, ...A_n, B_1, B_2, ...B_m)$ *in that order* [34]. Operators such as union do not preserve the order of tuple elements. As we shall see in Section 4.4.3, preserving the order is important in constructing directed edges.

**Table 11:** First-order graph of people and their working locations

(a) $N$: Person Nodes

| **Node Label** | **Locale** |
|---|---|
| John, Smith | $R_{\mathsf{E}}$.t1 |
| Franklin, Wong | $R_{\mathsf{E}}$.t2 |
| Jennifer, Wallace | $R_{\mathsf{E}}$.t3 |
| Ahmad, Jabbar | $R_{\mathsf{E}}$.t4 |

(b) $M$: Location Nodes

| **Node Label** | **Locale** |
|---|---|
| Los Angeles, CA | $R_{\mathsf{D}}$.t1 |
| San Jose, CA | $R_{\mathsf{D}}$.t2 |
| Houston, TX | $R_{\mathsf{D}}$.t3 |

(c) $E(N, M)$: Edges

| **n** | **m** | **Basis** |
|---|---|---|
| John, Smith | San Jose, CA | $R_{\mathsf{E}}.t1 + R_{\mathsf{D}}.t2$ |
| Franklin, Wong | Houston, TX | $R_{\mathsf{E}}.t2 + R_{\mathsf{D}}.t3$ |
| Jennifer, Wallace | Houston, TX | $R_{\mathsf{E}}.t3 + R_{\mathsf{D}}.t3$ |
| Ahmad, Jabbar | Los Angeles, CA | $R_{\mathsf{E}}.t4 + R_{\mathsf{D}}.t1$ |

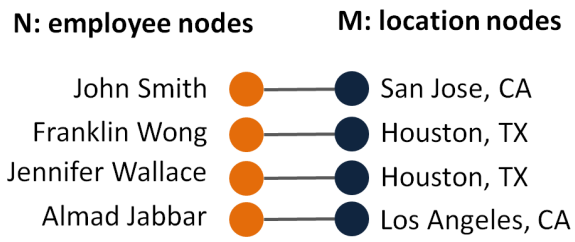



**Figure 26:** First-order graph of employees and locations

#### *4.2.2.2 Creating nodes and edges across multiple tables*

It is possible to construct first-order nodes whose labels and attributes are from different relations too. For example, we may want to construct nodes representing all the PIs (Principal Investigators) from Table 12(b). The node labels will come from the Name dimension in the PERSON relation, and each node will have an attribute called Role, based on the Role dimension in the PI – OF relation. Again we need to join the tables involved using a *left-outer-join* because we want to preserve all the node labels even when there are no matching attributes. The condition for the left outer join in this case is $R_{\mathsf{P}}.\mathsf{PID} = R_{\mathsf{W}}.\mathsf{PID}$ & $R_{\mathsf{W}}.\mathsf{Role} =$ "PI". We will have the PI nodes in Table 13(a). Note that the locale of the nodes is determined by only the table from which the labels are constructed.

Summarizing the above discussion, we arrive at the following formal definition:

**Definition 9.** [**First-order Attributed Node from Multiple Relations**] A first-order node $n$ can have its label $n.l$ constructed from a tuple $t_l$ in a relation $R_l$ and attribute $a$ constructed from a tuple $t_a$ in a different relation $R_a$ on condition $\theta$:

If $t_l \in R_l$ and $t_a \in R_a$ and $R_l.t_l + R_a.t_a \in R_l \bowtie_\theta R_a$,

$$\begin{cases} n.l = t_l.\Delta_l \text{ or } f(t_l.\Delta_l) \\ n.a = t_a.\Delta_a \text{ or } f(t_a.\Delta_a) \end{cases}$$

The locale of the node constructed will be $R_l.t_l$.

We can then construct another set of nodes representing program managers from

**Table 12:** Researchers and the grants they receive. GID represents Grant ID, and PID represents PI ID.

(a) $R_{\mathsf{G}}$ : GRANT

| GID | Title | Program | Program Manager | Amount | Year |
|---|---|---|---|---|---|
| 1 | Cognitive and Social Design of Robotic Assistants | Information Technology Research | William Bainbridge | 3325900 | 2001 |
| 2 | Subgroup Fault Lines in Distributed International Teams | ITR Small Grants | Ephraim P. Glinert | 205000 | 2002 |

(b) $R_{\mathsf{P}}$ : PERSON

| PID | Name | Org |
|---|---|---|
| 1 | Pamela Hinds | Stanford University |
| 2 | Jodi Forlizzi | Carnegie-Mellon University |
| 3 | Sara Kiesler | Carnegie-Mellon University |

(c) $R_{\mathsf{W}}$ : PI – OF

| PID | GID | Role |
|---|---|---|
| 1 | 1 | coPI |
| 2 | 1 | coPI |
| 3 | 1 | PI |
| 1 | 2 | PI |

$R_{\mathsf{G}}$ (Table 3a), shown in Table 13(b). Finally, we can connect these two types of first-order nodes following Definition 8, on the condition that $R_{\mathsf{G}}.\mathsf{GID} = R_{\mathsf{W}}.\mathsf{GID}$ & $R_{\mathsf{P}}.\mathsf{PID} = R_{\mathsf{W}}.\mathsf{PID}$. The edges are shown in Table 13(c). The graph is visualized in Figure 27.

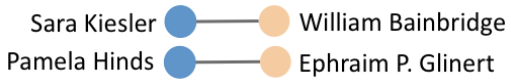



**Figure 27:** First-order graph of PIs and program managers

### 4.2.3 Slice 'n dice

When we create a set of nodes from a dimension list in one take, sometimes we want to specify conditions that imply certain perspectives on the graph construction. For example, in creating a graph connecting grant titles and researchers from the NSF dataset, we may want to only look at the patterns of researchers receiving grants with amount less than $500,000$ in year 2003 only. In this case, Amount and Year serve as slicing and dicing dimensions to divide a grant-researcher graph into subgraphs. The notion of slicing and dicing here is important as in the traditional relational

**Table 13:** First-order graph connecting PIs and the program managers

(a) $N$: PI Nodes

| Node Label | Role | Locale |
|---|---|---|
| Sara Kiesler | PI | $R_{\mathsf{P}}$.t3 |
| Pamela Hinds | PI | $R_{\mathsf{P}}$.t1 |

(b) $M$: ProgManager Nodes

| Node Label | Locale |
|---|---|
| William Bainbridge | $R_{\mathsf{G}}$.t1 |
| Ephraim P. Glinert | $R_{\mathsf{G}}$.t2 |

(c) $E(N, M)$: Edges

| n | m | Basis |
|---|---|---|
| ("Sara Kiesler", PI) | William Bainbridge | $R_{\mathsf{P}}.t3 + R_{\mathsf{G}}.t1$ |
| ("Pamela Hinds", PI) | Ephraim P. Glinert | $R_{\mathsf{P}}.t1 + R_{\mathsf{G}}.t2$ |

OLAP, and is akin to the drill-down operation in the informational dimensional Graph OLAP [24]. In the informational dimensional Graph OLAP, the drill-down operation generates snapshots that are different observations of the same underlying network.

In our framework, slice 'n dice is conceived as filtering conditions on tuples. Formally, to create a set of nodes $N$ from dimension list $\Delta_n$ with slicing and dicing dimension $\Delta_s$:

$$\forall n \in N, n.l \in \{t.\Delta_n \mid \mathsf{COND}(t.\Delta_s)\}$$

When $\Delta_n$ and $\Delta_s$ are from different tables, if the tables involved are independent, the condition has no filtering effect; otherwise, the tables are joined by a user-defined JOIN condition. Filtering conditions based on slicing and dicing dimensions typically restrict the locale of nodes constructed to a subset, and the edges are again established based on definitions 6 and 8.

## *4.3 Transformations: Higher-order Graphs*

First-order graphs may not be at the right level of abstraction for exploration and analysis. For example, in the first-order graph shown in Figure 25, there are duplicated nodes that may refer to the same real-world entities. Different transformations can be applied to first-order or higher-order graphs to make them more meaningful. In our formalism, we present three generic transformations: node aggregation,

projection and edge weighting.

### 4.3.1 Node Aggregation

Multiple ways of aggregating nodes (and hence edges) in a graph exist. While it is common practice to aggregate nodes of the same type, it is also possible that nodes from different types can be aggregated, depending on analytical needs. In general, node aggregation transforms a given graph $G = (N, E)$ into a new graph with new sets of nodes and edges: $G' = (N', E')$. Given a function $\mathscr{G}$ describing how nodes should be aggregated: $n' = \mathscr{G}(n_1, \ldots, n_j)$ where $\{n_1, \ldots, n_j\} \subset N$ and $n' \in N'$, the new node's locale are the union of the original nodes' locales: $locale(n') = locale(n_1) \cup \ldots \cup locale(n_j)$.

Given that a set of nodes $A = \{a_1, a_2, \ldots, a_j\}$ are aggregated into one node $a'$, and that a set of nodes $B = \{b_1, b_2, \ldots, b_k\}$ are aggregated into one node $b'$, the edge between $a'$ and $b'$, if any, will be derived from any edge between the original nodes:

$$basis(a', b') = \bigcup \{basis(a, b) \mid a \in A \wedge b \in B\}$$

If $basis(a', b') = \emptyset$, $a'$ and $b'$ are not connected.

These two specifications define various types of node aggregation. A typical example is **entity resolution**. Two nodes may be considered to refer to the same logical entity if they have identical labels, or if they have identical labels and attributes, or through a more sophisticated entity resolving algorithm [20]. From the graph in Table 9 (visualized in Figure 25) , we can aggregate nodes if they share identical labels and attributes. This gives us the following graph in Table 14, and the process of entity-resolution is visualized in Figure 28.

**Table 14:** A higher-order graph obtained by resolving entities

(a) $N$: Person Nodes

| n | locale |
|---|---|
| (“Dodd, Chris”, VA) | t1 |
| (“Smith, John”, VA) | t2 |
| (“Smith, John”, AL) | t3 |
| (“Hirani, Amyn”, VA) | t4 |
| (“Keehan, Carol”, VA) | t5, t6 |

(b) $M$: Location Nodes

| m | locale |
|---|---|
| WH | t1, t2, t4, t5 |
| OEOB | t3, t6 |

(c) $E(N, M)$: Edges

| n | m | basis |
|---|---|---|
| (“Dodd, Chris”, VA) | WH | t1 |
| (“Smith, John”, VA) | WH | t2 |
| (“Smith, John”, AL) | OEOB | t3 |
| (“Hirani, Amyn”, VA) | WH | t4 |
| (“Keehan, Carol”, VA) | WH | t5 |
| (“Keehan, Carol”, VA) | OEOB | t6 |

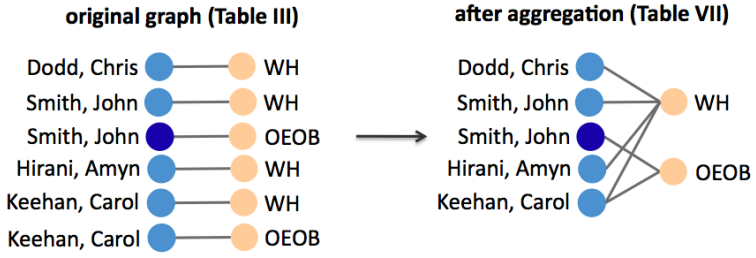



**Figure 28:** Nodes in a first-order graph are aggregated if they have the same labels and attributes. Here we have two “Smith, John” nodes with different attribute values encoded by color, after entity resolution, they are treated as different entities.

Another type of node aggregation is **attribute-based** aggregation or **pivoting**. PivotGraph [96] terms this operation *roll-up* based on traditional OLAP terminology. From the graph presented in Table 14, we can aggregate people nodes by attribute Type and obtain the graph in Table 15. The resulting graph shows the locations that are typically visited for different types of visits. The pivoting process is visualized in Figure 29.

**Table 15:** A higher-order graph obtained by pivoting

(a) $N$: Type Nodes

| n | locale |
|---|---|
| VA | t1, t2, t4, t5, t6 |
| AL | t3 |

(b) $M$: Location Nodes

| m | locale |
|---|---|
| WH | t1, t2, t4, t5 |
| OEOB | t3, t6 |

(c) $E(N, M)$: Edges

| n | m | basis |
|---|---|---|
| VA | OEOB | t6 |
| VA | WH | t1, t2, t4, t5 |
| AL | OEOB | t3 |

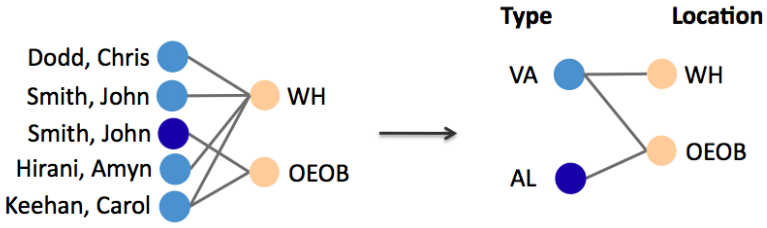



**Figure 29:** Nodes in a higher-order graph are aggregated by their attributes

### 4.3.2 Projection

All the graphs that can be constructed so far are *multi-mode* or *n-partite*: there are multiple types of nodes, and edges exist only between nodes of different types. In social network analysis, it is often necessary to transform a multi-mode graph into a one-mode graph (i.e. there is only one type of node) because many analytical metrics are designed for one-mode graphs only. Projection is a commonly used technique for transforming a 2-mode graph into a 1-mode graph [61].

Projection is a process where two nodes of the same type are connected if they are both connected to the same node of another type. For example, in a two mode $\mathsf{LastNm}, \mathsf{FirstNm} - \mathsf{Loc}$ graph shown on the left of Figure 30, $\mathsf{Smith}, \mathsf{John}$ and $\mathsf{Keehan}, \mathsf{Carol}$ both are connected to $\mathsf{OEOB}$. In the one-mode projection they are then connected. The basis of the edge between these two people is no longer a set of tuples, but a set of nodes (e.g. OEOB). The projection process is visualized in Figure 30. An

interpretation of this graph is that two persons are connected if they have visited the same location.

**Table 16:** A higher-order one-mode graph obtained by projection

(a) $N$: Person Nodes

| **n** | **locale** |
|---|---|
| Dodd, Chris | t1 |
| Smith, John | t2, t3 |
| Hirani, Amyn | t4 |
| Keehan, Carol | t5, t6 |

(b) $E(N_1, N_2)$: Edges

| $n_1$ | $n_2$ | **basis** |
|---|---|---|
| Dodd, Chris | Smith, John | WH |
| Dodd, Chris | Hirani, Amyn | WH |
| Dodd, Chris | Keehan, Carol | WH |
| Smith, John | Hirani, Amyn | WH |
| Smith, John | Keehan, Carol | WH, OEOB |
| Hirani, Amyn | Keehan, Carol | WH |

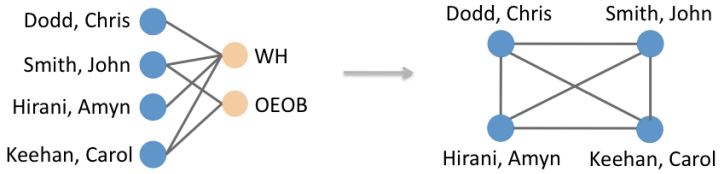



**Figure 30:** Transforming a two-mode graph into one-mode by projection

Formally, for a graph $G = (N, M, E)$ with two types of nodes $N$ and $M$, the $M$-projection of $G$ is the graph $G' = (N, E')$. Two nodes $n_1, n_2 \in N$ are connected if they have at least one neighbor in common in $M$: $\exists m \in M,\ e(n_1, m) \in E \wedge e(n_2, m) \in E \Rightarrow e(n_1, n_2)$. According to this definition, the basis of an edge is no longer a set of tuples, but a set of nodes: $basis(n_1, n_2) = \{m \in M \mid e(n_1, m) \in E \wedge e(n_2, m) \in E\}$.

We can also extend projection to tri-mode graphs. In Figure 31, we have a graph with three types of nodes: researchers, program and program managers. With projection, a researcher and a program manager are connected if they are both connected to the same program. In the resulting graph, researchers and program managers are thus linked by the research programs.

Formally, the $L$-projection of a tri-mode graph $G = (L, M, N, E)$ is a graph $G' = (M, N, E')$. A node $m \in M$ and a node $n \in N$ are connected if they have at least one common neighbor (i.e. adjacent node) in $L$: $\exists l \in L,\ e(m,l) \in E \wedge e(n,l) \in E \Rightarrow e(m,n)$. If there are edges connecting $n$ and $m$ that are independently constructed from the projection process, the graph will be multi-plex (i.e. the graph contains edges of different types).

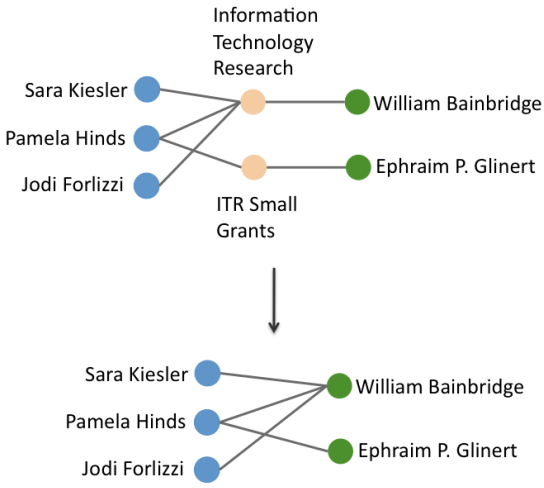



**Figure 31:** Transforming a tri-mode graph into a two-mode one by projection

### 4.3.3 Edge Weighting

A numerical weight can be assigned to an edge indicating the strength of connection between a pair of nodes. By default, the cardinality of the basis can be assigned as an edge's weight: $w(n_1, n_2) = |basis(n_1, n_2)|$. When the basis of an edge is a set of tuples, the edge weight is essentially the frequency of co-occurrence. When the basis of an edge is a set of nodes, the edge weight is the number of common nodes projected.

When the basis of an edge is a set of tuples, we can enrich the semantics of edge weight by using aggregation of specific tuple dimension values: $w(n_1, n_2) = \mathscr{F}(t.\Delta)$ where $t \in basis(n_1, n_2)$. $\mathscr{F}$ stands for aggregation operations (see [34], Chapter 7), which include typical functions such as COUNT, SUM, AVERAGE, MAXIMUM and MINIMUM. $w(n_1, n_2) = |basis(n_1, n_2)|$ hence is equivalent to $w(n_1, n_2) = \mathscr{F}_{\mathsf{COUNT}}(t.\mathsf{ID})$

where ID is the unique identifier.

From the graph presented in Table 15, we can define the weight of an edge to be the sum of group sizes of each tuple in the edge's basis: $w(n_1, n_2) = \Sigma\ t.\text{GroupSize}$. This gives us the graph in Table 17, visualized in Figure 32, where the edge weights can be interpreted as the number of people involved in each visit type to each location.

**Table 17:** A higher-order graph obtained by assigning attribute-based edge weights

(a) $N$: Type Nodes

| n | locale |
|---|---|
| VA | t1, t2, t4, t5, t6 |
| AL | t3 |

(b) $M$: Location Nodes

| m | locale |
|---|---|
| WH | t1, t2, t4, t5 |
| OEOB | t3, t6 |

(c) $E(N, M)$: Edges

| n | m | basis | weight |
|---|---|---|---|
| VA | OEOB | t6 | 26 |
| VA | WH | t1, t2, t4, t5 | 2447 |
| AL | OEOB | t3 | 144 |

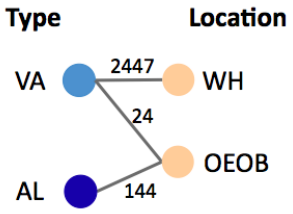



**Figure 32:** Assigning GroupSize as the Edge Weight

## *4.4 Expressive Power*

In this section we discuss two examples to show how different operations presented in the previous section can be combined to produce graphs with desired semantics.

### 4.4.1 Proximity-based Graph Construction

A number of transformations can be combined to construct a graph with desired semantics. For example, we may want to look at the network where visitors are

connected if their visits are within a 3-day range. To construct this graph, we first create a first-order graph with two types of nodes: $\langle \mathsf{LastNm}, \mathsf{FirstNm} \rangle$ and $\langle \mathsf{Date} \rangle$, and then apply node aggregation to these two types of nodes: for people nodes, two nodes are aggregated into one if they have identical labels; for date nodes, two nodes are aggregated into one if their time difference is within 3 days, and the aggregated node's label is a concatenation of the labels of the two original nodes. Table 18 shows the nodes and edges in the graph after proximity-based aggregation of $\langle \mathsf{Date} \rangle$ nodes.

**Table 18:** A graph connecting White House Visitors with their visiting dates, the dates are aggregated based on proximity

(a) $N$: People Nodes

| Node Label | locale |
|---|---|
| Dodd, Chris | t1 |
| Smith, John | t2, t3 |
| Hirani, Amyn | t4 |
| Keehan, Carol | t5, t6 |

(b) $M$: Date Nodes

| Node Label | locale |
|---|---|
| 6/25/09 - 6/26/09 | t1, t2 |
| 6/26/09 - 6/29/09 | t2, t3 |
| 6/29/09 - 6/30/09 | t3, t4 |
| 6/29/09 - 7/1/09 | t3, t5 |
| 6/30/09 - 7/1/09 | t4, t5 |

(c) $E(N, M)$: Edges

| n | m | basis |
|---|---|---|
| Dodd, Chris | 6/25/09 - 6/26/09 | t1 |
| Smith, John | 6/25/09 - 6/26/09 | t2 |
| Smith, John | 6/26/09 - 6/29/09 | t2, t3 |
| Smith, John | 6/29/09 - 6/30/09 | t3 |
| Smith, John | 6/29/09 - 7/1/09 | t3 |
| Hirani, Amyn | 6/29/09 - 6/30/09 | t4 |
| Hirani, Amyn | 6/30/09 - 7/1/09 | t4 |
| Keehan, Carol | 6/29/09 - 7/1/09 | t5 |
| Keehan, Carol | 6/30/09 - 7/1/09 | t5 |

**Table 19:** A graph showing connections between White House visitors. Two visitors are connected if their visit dates are within 3 days to each other.

(a) $N$: People Nodes

| **Node Label** | **locale** |
|---|---|
| Dodd, Chris | t1 |
| Smith, John | t2, t3 |
| Hirani, Amyn | t4 |
| Keehan, Carol | t5, t6 |

(b) $E(N_1, N_2)$: Edges

| $n_1$ | $n_2$ | **basis** | **weight** |
|---|---|---|---|
| Dodd, Chris | Smith, John | 6/25/09 - 6/26/09 | 1 |
| Smith, John | Hirani, Amyn | 6/29/09 - 6/30/09 | 1 |
| Smith, John | Keehan, Carol | 6/29/09 - 7/1/09 | 1 |
| Hirani, Amyn | Keehan, Carol | 6/30/09 - 7/1/09 | 1 |

We then do a projection on Date and get the following one-mode graph, and we define the edge weight to be the cardinality of the basis of the edge. An interpretation of this graph (Table 19) is that visitors are linked by the proximity of their visit times, and the connection strength indicates how frequent their visits are close in time to each other.

### 4.4.2 Subtleties in Edge Semantics

When we create edges between two types of nodes from different tables, the method used in constructing connections will affect the numerical weights assigned to the edges and how the edges are interpreted. For example, we can directly connect Program Manager nodes from Table 3(a) and Org nodes from Table 3(b), and the meaning of connection is that of managers granting awards to organizations. The exact meaning of the edge weight, however, is more subtle. If we join Table 3(a) and Table 3(b) on the condition that GRANT.GID = WORKON.GID and PERSON.PID = WORKON.PID, the edges between program managers and organizations will have the semantics of Program Manager – GIDxPID – Org. The edge between William Bainbridge and Carnegie Mellon University, for example, will have a weight of 2, indicating that he has awarded grants to researchers from this organization twice (to Jodi Forlozzi once and to Sara Kiesler

once). That is, both the number of researchers per grant and the number of grants will have an impact on the edge weight.

This weight however may not be at the right level of abstraction to the analyst, as Jodi Forlozzi and Sara Kiesler have collaborated on the same grant, and the program manager has in fact only awarded one grant to the organization. To let the weight reflect the number of unique grants awarded by the program manager to the organization only, we can connect Program Manager and GID explicitly first, then connect GID with Orgs. We then do a projection by connecting a Program Manager with an Org if they both connect to the same GID. The weight assigned to the edge between William Bainbridge and Carnegie Mellon University will then be 2, indicating two grants.

These subtleties of edge construction reinforce the notion that we can create connections between nodes with great flexibility and rich semantics. A program manager and an organization, for example, can be connected by the grants awarded by the manager to the organization, by the frequency of awards to researchers from this organization, or by the researchers from the organization who receive grants from the manager, to name a few.

### 4.4.3 Constructing One-Mode Directed Graphs

In Section 4.4.1, we have demonstrated how we can construct a one-mode graph using the projection operator. The graph created is undirected because the concept of proximity does not have any semantics of direction. In many examples provided earlier where two-mode graphs are created (e.g. the visitor-location graph in Figure 28), explicit declaration of edge direction is not essential either: as we have discussed in Section 4.1, the existence of distinct types of nodes implies the possibility of interpreting the edges as directed.

There are cases, however, where edges need to be explicitly labeled as directed. Consider the example in Table 4, the semantics of the connections between two

persons have the explicit notions of direction: the from column indicates source and the to indicates target. We thus need to discuss how our framework can be applied to construct directed graphs from such data tables describing reflexive relationships.

Suppose we want to construct a graph showing the email communication patterns between different departments. We first create two identical sets of first-order Department nodes from Table 4(a). The resulting nodes are shown in Table 20 (a) and (b). To construct connections between these two sets of first-order nodes, we follow Definition 8. A node $n \in N$ is connected to a node $n_1 \in N_1$ if $locale(n) + locale(n_1) \in R_{\mathsf{P}} \bowtie R_{\mathsf{P}}$ on the condition that $R_{\mathsf{P}}.\mathsf{ID} = R_{\mathsf{E}}.\mathsf{From}$ and $R_{\mathsf{P}}.\mathsf{ID} = R_{\mathsf{E}}.\mathsf{To}$. This gives us the edges in Table 20 (c). Since we have used the concatenation operator in Definition 8, we have ensured that the order of tuple values is preserved in edge creation, we can thus assign directions to edges in Table 20 (c) by specifying $n$ nodes as source nodes and $n_1$ nodes as target nodes. Finally, we can do an aggregation of $n$ and $n_1$ nodes if they share the same label. Figure 33 shows the final graph.

**Table 20:** Constructing a directed one-mode graph from data tables describing reflexive unary relationships

(a) $N$: People Nodes

| **Node Label** | **locale** |
|---|---|
| Sales | $R_{\mathsf{P}}$.t1 |
| Marketing | $R_{\mathsf{P}}$.t2 |
| Engineering | $R_{\mathsf{P}}$.t3 |
| Research | $R_{\mathsf{P}}$.t4 |
| Marketing | $R_{\mathsf{P}}$.t5 |

(b) $N_1$: People Nodes 1

| **Node Label** | **locale** |
|---|---|
| Sales | $R_{\mathsf{P}}$.t1 |
| Marketing | $R_{\mathsf{P}}$.t2 |
| Engineering | $R_{\mathsf{P}}$.t3 |
| Research | $R_{\mathsf{P}}$.t4 |
| Marketing | $R_{\mathsf{P}}$.t5 |

(c) $E(N, N_1)$: Edges

| $n$ | $n_1$ | **Basis** |
|---|---|---|
| Sales | Research | $R_{\mathsf{P}}.t1 + R_{\mathsf{P}}.t4$ |
| Marketing | Sales | $R_{\mathsf{P}}.t2 + R_{\mathsf{P}}.t1$ |
| Engineering | Marketing | $R_{\mathsf{P}}.t3 + R_{\mathsf{P}}.t2$ |
| Research | Sales | $R_{\mathsf{P}}.t4 + R_{\mathsf{P}}.t1$ |
| Marketing | Marketing | $R_{\mathsf{P}}.t5 + R_{\mathsf{P}}.t2$ |

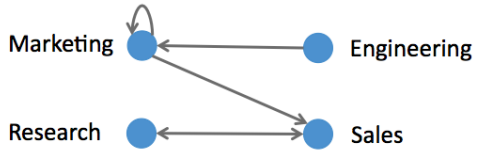



**Figure 33:** A directed graph showing email communications between different departments in a company

### 4.4.4 Potential Limitations

In working with sample datasets we have already identified situations that point to potential limitations of the current framework. For example, if we want to create a network where two organizations are connected if they have collaborated on more than two grants within the past five years, the set of operations in the current framework is not sufficient to express such semantics of conditional connectivity.

Further work is required to understand the expressive power and limitations of this framework. Relational algebra, an established framework, is proven to be equivalent to first-order logic, and the expressive power of first-order logic is well understood [34]. In relational algebra, a set of primitive operators serves as building blocks for more complex operators. Since we are investigating a new domain here, it remains to be seen if the set of operations we defined over one or more tables can serve as primitives for graph construction and if any additional operations need to be included for completeness.

# CHAPTER V

# PLOCEUS: DESIGN FOR NETWORK-BASED VISUAL ANALYSIS

## *5.1 System Design*

In the previous chapter, we presented a detailed treatment of the formal framework that provides analytical power in data transformation and graph construction. This framework can serve as the foundation of a variety of different system and interface designs. For example, we can take a toolkit approach by providing programming interfaces to coders, similar to the design of prefuse [48]. A toolkit is not always the best system for fast and seamless exploratory analysis however: analysts must be able to code, and more importantly, the stage of modeling is separated from the stage of visual exploration.

To design for a more seamless experience, the system must provide an integrated environment for analysts to perform network modeling and to receive immediate visual feedback. Ideally the interface will also support smooth transitions between the phases of network modeling, visual exploration and in-depth drill down.

These goals have two major implications/requirements on our system and interface design: first, we need to provide a seamless experience that unifies data transformation with visual exploration and analysis; second, the formal and operational mechanisms should be hidden from the users and abstracted as intuitive interaction techniques. Figure 34 shows the desired system architectural design.

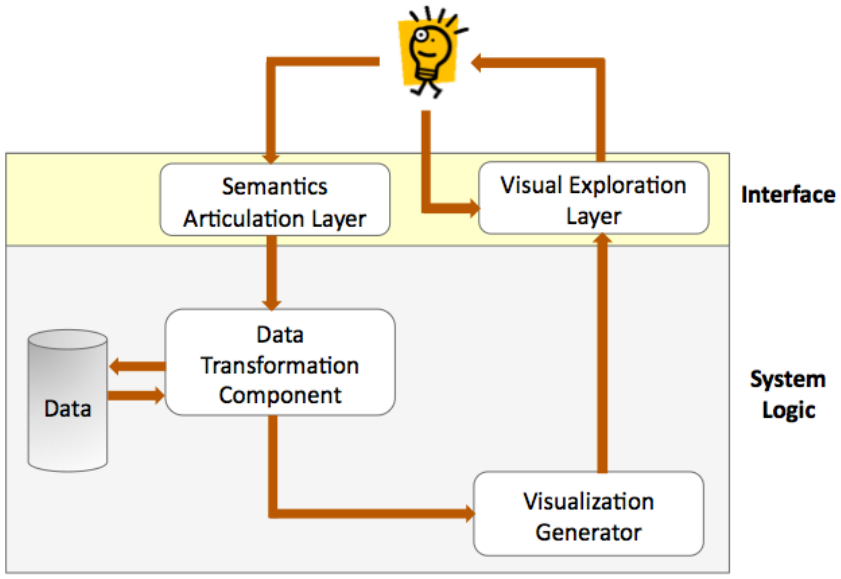



**Figure 34:** The overall architecture of the system design.

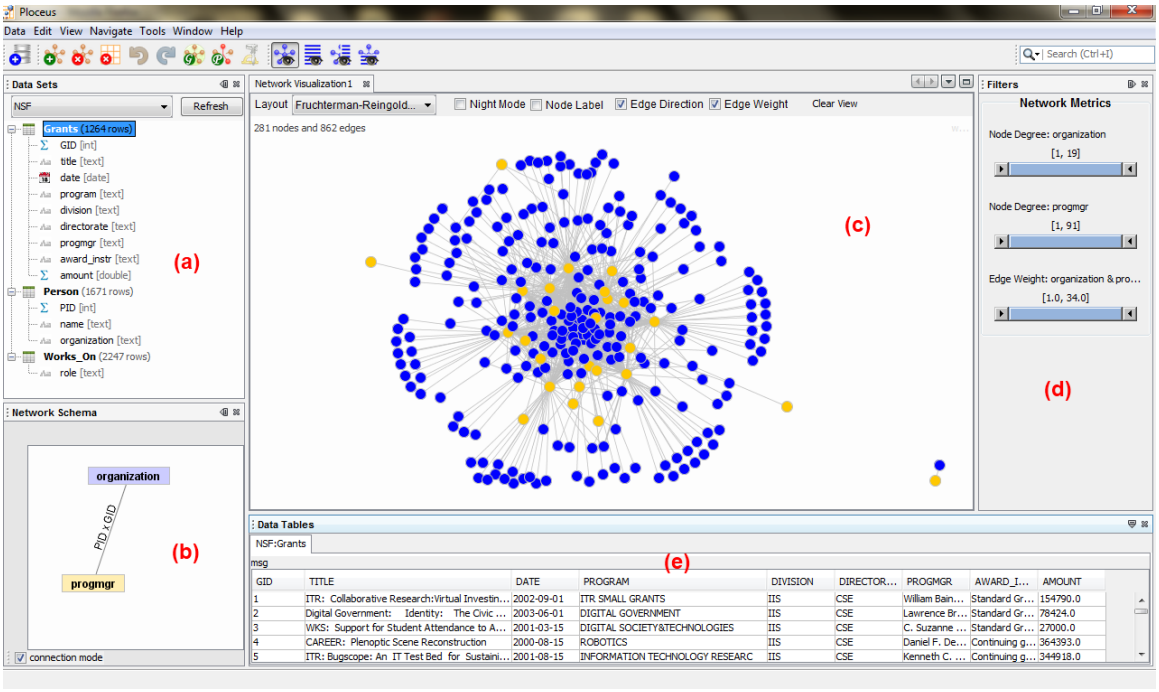



**Figure 35:** Five window components in the Ploceus system interface: (a) data table column view on the top left, (b) network schema view on the bottom left, (c) network visualization view in the center, (d) filter panel on the right, and (e) data table viewer on the bottom.

Based on this architectural sketch, we have designed and implemented the system

Ploceus [1]. To address the first requirement, we provide an integrated environment consisting of multiple coordinated views. Ploceus has five window components (Figure 35). The data management view displays data tables users have imported and the columns within each table. The network schema view is a sandbox-like environment where users can construct and manipulate networks at a conceptual level, and shows a schema representation of the network users are currently modeling and analyzing. The network visualization view is where visual exploration and analysis take place, and shows immediate visual feedback on the network being created and updates whenever the network schema is modified. Users can create as many networks as they wish, and each network is shown in a tab. When users switch between the tabs, the network schema view will update accordingly to reflect the schema of the network they are currently examining. The filter panel provides a set of range sliders to filter nodes and edges shown in the visualization. Finally the data table viewer allows you to look at the raw data tables in the traditional form of rows and columns. Users can directly drag and drop a table item (e.g. "Grants" in Figure 35) from the data management view to the data table view to load all the data in that table, or they can load the data through a context-menu associated with table items. Each of these window components can be minimized, maximized, resized, docked or undocked.

To address the second requirement, Ploceus provides a direct manipulation interface, combined with user dialogs, for fast construction and transformation of networks. Through simple mouse gestures such as drag and drop, users can perform model construction and manipulation in a visual manner. In the following sections, we discuss the designs and implementation issues in building Ploceus.

[1] In nature, Ploceus is a kind of weaver birds that can build sophisticated nests.

## 5.2 Semantics Articulation

### 5.2.1 Design Choices

Ploceus currently supports the following types of operations. In this section, we describe these operations at a functional level and their relationships to the formal framework discussed in Chapter 4.

The major design challenge in building Ploceus is *how to reduce articulatory distance*, i.e. assuming the analysts want to perform some operations, what is an intuitive way for them to communicate the intent to the system.

One possible design is to integrate a visual interface with a scripting interface as done in GUESS [13] - the manipulation of graphs is in the form of commands (Figure 36). The advantage of this approach is that script languages are precise, expressive and concise; the disadvantage of this approach is that analysts must understand basic programming concepts.

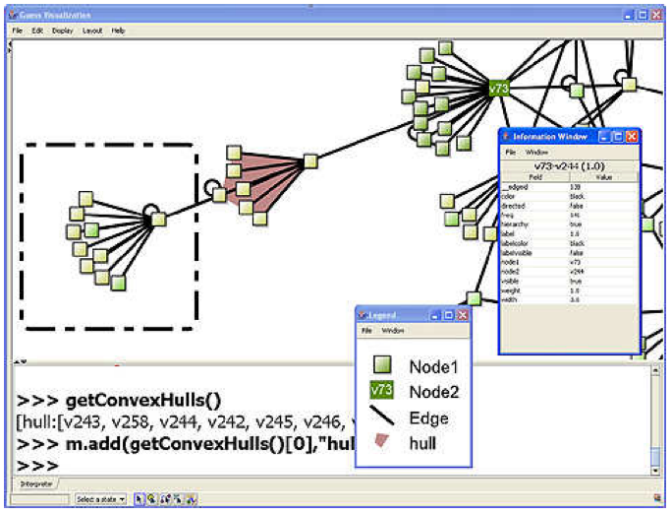



**Figure 36:** The GUESS system interface. Users specify interaction with the network visualization through a scripting window at the bottom. Here the user is printing the nodes in a convex hull (a set of nodes) and adding the first node in the hull to another set

Another design alternative is Programming by Demonstration (PBD) [29]. PBD typically uses a direct manipulation interface. For example, to create connections between LName, FName nodes and Loc nodes constructed from Table 1, assuming we

already have a visual representation of the existing nodes, we can use a click-drag-drop mouse gesture to connect a visitor node (e.g. "Dodd, Chris" ) and a location node (e.g. "WH" ). After we have performed similar gestures two or three times, the system will figure out that our intention is to connect LName, FName nodes and Loc nodes, and will perform the same operation automatically on the rest of the nodes.

PBD arguably shortens articulatory distance when it works on a direct manipulation interface. As shown in the create-connection example, users perform the exemplary operation at the level of individual data items, and the system generalizes from the user interaction to a high level by connecting different types of nodes. This bottom-up design approach has some shortcomings for network modeling. Analysts need to know if an edge indeed exists between two specific nodes, and they thus need to access a low level representation of the raw data and understand the mechanism of edge computation.

### 5.2.2 Direct Manipulation Interface

Our final design decision is to adopt a direct manipulation approach akin to that of Polaris [84] and Tableau [6]. Instead of PBD, analysts directly interact with high-level conceptual representations of the relational data schema and indicate intention by manipulating these representations. In the section, we present detailed interaction designs for various construction and transformation operations.

- **Create Nodes**: Transform the values in one or more columns into node labels. For example, we can construct a set of nodes representing the people visiting the White House from all the rows in Table 1, and the labels of the nodes are created from the LName and FName columns. This results in four nodes: "Dodd, Chris", "Smith, John", "Hirani, Amyn", and "Keehan, Carol". This operation draws from Definition 4 and the entity resolution process in Section 4.3.1, where we first create a set of first-order nodes and aggregate them if they share the

same label. In the Ploceus interface, this operation is implemented as a simple drag-and-drop interaction: analysts create nodes by dragging selected table columns in the data management view to an empty area in the network schema view. Each drag-and-drop action creates a type of node, and the system assigns a color to that type. (Figure 37).

- **Add Attributes**: Transform the values in one or more columns as attributes of existing nodes. For example, we can add an attribute AccessType to the people nodes constructed from LName, FName earlier. The node "Dodd, Chris" will have the value "VA" for the AccessType attribute. Ploceus supports adding columns as attributes from a different table too. For example, we can add Role from Table 3(c) as an attribute for the Name nodes constructed from Table 3(b). Ploceus only allows a node to have one value for any particular attribute, so there will be two "Sharad Mehrotra" nodes in this case, one having a PI role and the other having a CoPI role. This operation draws on Definition 4 and Definition 9. Dragging and dropping columns on top of an existing node type add those columns as an attribute to the node type 38.

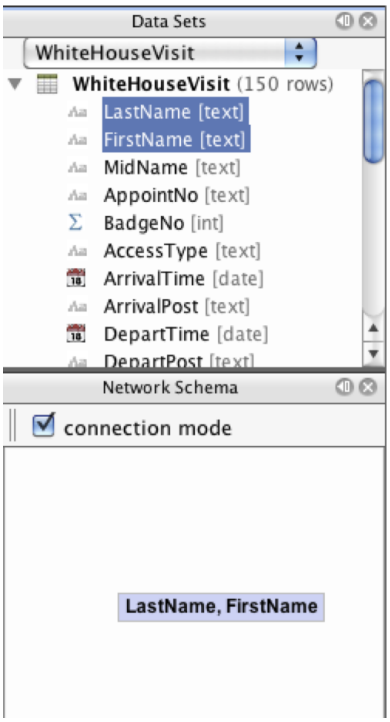



**Figure 37:** Analysts create nodes in Ploceus by dragging selected table columns from the data management view to an empty area in the network schema view

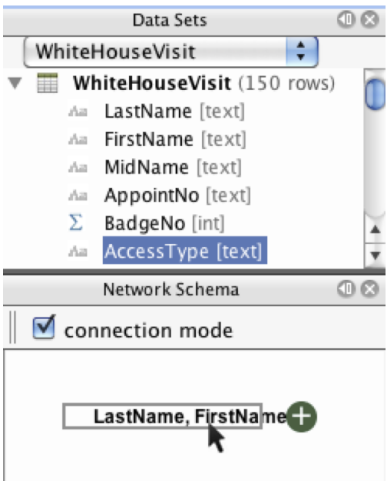



**Figure 38:** Analysts add node attribute in Ploceus by dragging selected table columns from the data management view on top of existing node types in the network schema view

- **Create Connections**: Create edges between existing nodes. This operations is based on Definition 6, 8, and the discussion of edge aggregation in Section 4.3.1. For example, we can connect LastNm, FirstNm nodes and Loc nodes from Table 1 to see the visiting patterns by the visitors to the various locations. We can also connect nodes created from different tables, e.g. ProgramManager nodes from Table 3(a) and Org nodes from Table 3(b). Given two types of nodes, analysts create connections between them by clicking on one type of nodes and dragging the mouse to the other type of nodes in the network schema view (Figure 39). This action draws an edge between the two that takes effect when the mouse button is released.

  When multiple tables are involved, Ploceus determines how the tables should be joined by analyzing the foreign key constraints between the tables through the Dijkstra shortest-path algorithm [31]. In this case, the two tables are joined through Table 3(c). Ploceus computes whether there should be an edge between any two nodes as well as assigns a weight to that edge. When multiple ways of joining tables are possible, users can specify the join condition through a dialog.

  Given a set of diverse operations supported in Ploceus, it is important for analysts to correctly interpret the edge semantics in the networks created. As we have discussed in Section 4.4.2, different construction paths lead to different edge semantics. To help analysts keep track of what they are doing when connecting nodes from different tables, Ploceus labels the edge representation in the network schema view, indicating the semantics of the edges. Figure 41(a) shows the label for a progmgr – organization network where the edge weight between a program manager $p$ and an organization $o$ denotes the number of researchers from the organization $o$ who have received grants awarded by $p$, and Figure 41(b) shows the label for a progmgr – organization network where the edge weight represents the number of unique grants that a program manager has

given to an organization.

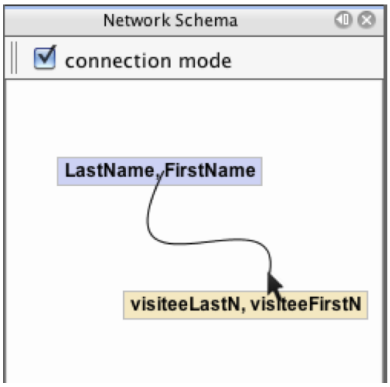



**Figure 39:** Analysts create connections in Ploceus by clicking on one type of nodes and dragging the mouse to the other type of nodes in the network schema view

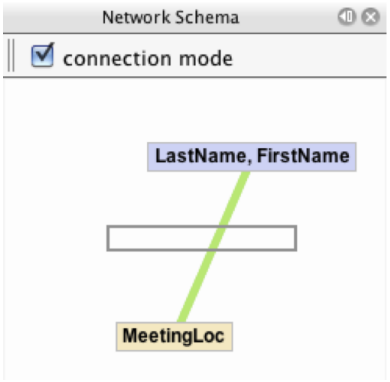
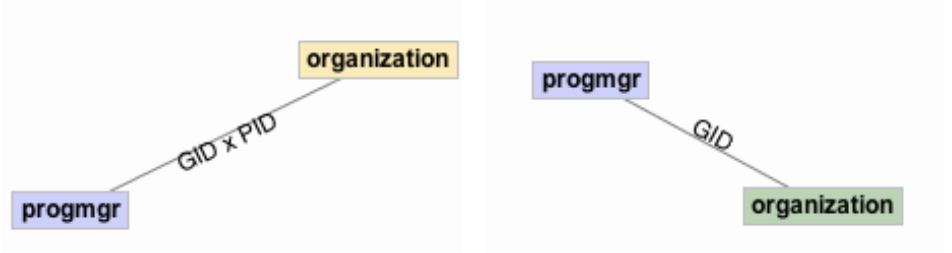



**Figure 40:** Analysts assign edge weight in Ploceus by dragging and dropping the column over the edge representation in the network schema view




(a) (b)

**Figure 41:** Edge semantics labels in the network schema view: a) the edge weight between a program manager $p$ and an organization $o$ denotes the number of researchers from the organization $o$ who have received grants awarded by $p$ , b) the edge weight represents the number of unique grants that a program manager has given to an organization

- **Assign Weights**: Assign numerical weights to edges. This operations is based on the discussion in Section 4.3.3. Ploceus by default assigns a weight to each edge created, indicating the frequency of co-occurrence between the nodes in the data . For example, if we connect LName, FName nodes and Loc nodes from Table 1, by default the edge between "Dodd, Chris" and WH has a weight of 1, indicating this person has visited the White House once in this dataset. We may instead want to represent the connection strength by the number of people he has brought on his visits, and assign the column Size as the edge weight. The edge between "Dodd, Chris" and WH will have a weight of 2018. Only a single column can be assigned as edge weight, and that column must be quantitative. To designate a quantitative column as edge weights, analysts drag and drop the column over the edge representation in the network schema view (Figure 40).

- **Project**: Connect two nodes if they both are connected to the same node of a different type. In a two-mode LName, FName - Loc network, for example, after projecting LName, FName nodes on Loc nodes, The weight of edges after projection reflects the unique number of Loc nodes being projected. Ploceus automatically computes the newly added edges with appropriate edge weights and removes the nodes being projected, in this case, the Loc nodes. Section 4.3.2 discusses the definition and mechanism of projection.

  Projection is a highly conceptual operation, and a natural-language like interface seems the most appropriate. We thus design the interface to be in the form of dialog interaction rather than drag and drop. To specify projection, analysts indicate through combo boxes the types of nodes to be projected (Figure 42). The dialog offers a preview of how the network will appear after projection, so that analysts can have a feel of the consequences of their actions.

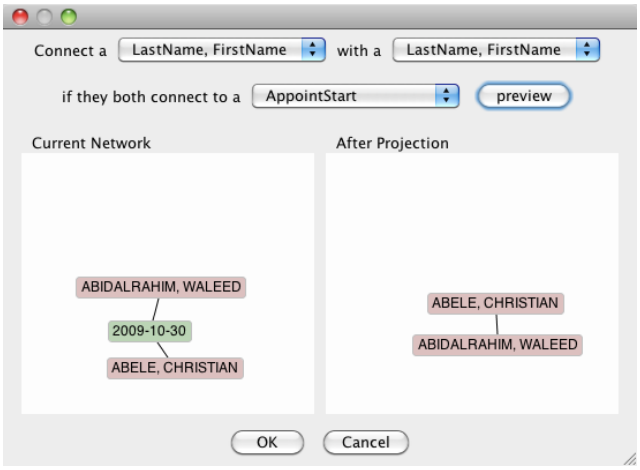



**Figure 42:** Analysts specify projection through dialogs with previews

- **Aggregate**: Group multiple nodes and treat them as one node. Ploceus automatically aggregates nodes with identical labels if no attributes are specified for these nodes, and aggregates nodes with identical labels and values if attributes are specified for the nodes. As a result, we have four distinct LName, FName nodes from Table 1, while there are actually six rows in the table.

  Ploceus supports following types of aggregation:

  - *Pivoting*: Section 4.3.1 discusses this aggregation in detail.
  - *Binning*: for nodes whose labels or attributes are derived from quantitative columns, value based aggregation is possible. One type of value based aggregation is *binning*: we divide the range from the minimum to the maximum attribute values into bins. For example, we can categorize Amount nodes created from Table 3(a) into three bins: "small" if Amount $<=$ 500k, "medium" if 500k $<$ Amount $<=$ 1200k, and "large" if Amount $>$ 1200k.
  - *Proximity grouping*: group nodes in a pair-wise manner if they have values close to each other. For example, from Table 2(b) we can create City nodes with attributes Latitude and Longitude. We can then aggregate every pair of City nodes into one for which the distance between them, computed from the

latitude and longitude information, is within 500 miles. This operation is combinatorial: if there are four cities, and every one is within 500 miles to each of the other three, proximity grouping will produce $\sum_{k=1}^{(4-1)} k = 6$ nodes. Proximity grouping is useful when combined with projection, so that we can, for example, create a network of employees whose workplaces are within 500 miles to each other (to do this, connect employee names with cities, aggregate cities, then project employees on cities).

Similar to projection, analysts specify aggregation operations through dialogs. They choose the type of aggregation through radio buttons (Figure 43). Depending on the properties of nodes selected, some operations may not be applicable. For example, when nodes have no attributes, pivoting does not make sense.

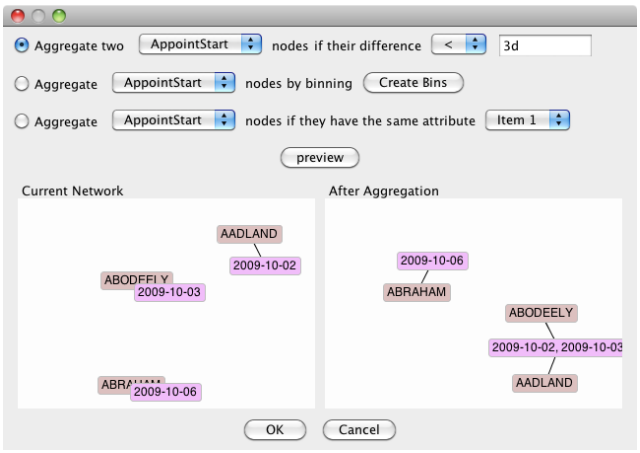



**Figure 43:** Analysts specify aggregation through dialogs with previews

- **Slice 'n Dice**: Divide a network into sub-networks based on selected columns. For example, given that we have constructed a LName, FName - Visitee network from Table 1, we may want to see how the visiting pattern is related to the locations of visits by dividing the network using Loc slices. We will then have two subnetworks, one representing the visiting patterns at the White House

("WH"), and the other at the Old Executive Office Building ("OEOB"). Slice 'n dice thus enables analysts to create and organize meaningful snapshots of a big network based on different perspectives. The values in columns used for slicing and dicing are either categorical or can be categorized. When hierarchical categories exist, analysts can slice and dice at multiple granularities, e.g. for a *date* column: day → week → month → quarter → year.

Ploceus supports slicing and dicing for up to 2 dimensions, designated as the horizontal and vertical axes in the visualization. Analysts specify the orientation of the slices (horizontal or vertical) by dropping columns to the appropriate shelf (Figure 44). A shelf is a horizontal or vertical axis in the network schema view.

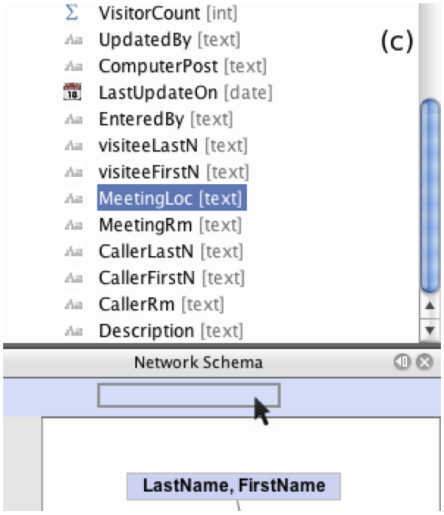



**Figure 44:** Analysts specify slice 'n dice dimensions by dropping columns to the appropriate shelf in the network schema view

## *5.3 Integrating Visual and Statistical Methods*

### 5.3.1 Node-link Layout and Network Matrix

To achieve tight coupling between network modeling and visual exploratory analysis, Ploceus provides coordination between different components of the interface. Whenever analysts perform an operation, the network view provides immediate feedback in the form of a node-link visualization of the current network (Figure 35). In

such node-link representations, Ploceus supports different layout algorithms such as Fruchterman-Reingold force-directed layout [39], circular layout, spring layout, the Kamada-Kawai algorithm [55] and Meyer's self-organizing layout [72].

Interactive features in node-link visualizations include interaction with individual nodes such as selecting, filtering, moving, hiding, showing, expanding (showing neighbors of a node) and displaying shortest paths between two nodes. Interaction with the visualization includes techniques of zooming and panning.

When the size of the network exceeds a threshold (currently defined as 750 nodes), to avoid screen clutter and low system performance, the node-link visualization will randomly sample and show a subpart of the network. In this case, the network view will show a text label on the top left indicating the total number of nodes and edges in the current network (Figure 45). If the nodes and edges that the analysts look for are not shown in this subnetwork, the analysts can clear the entire visualization (without modifying the underlying graph data structure), and interactively add selected nodes and edges to the visualization through a search query field on the top right corner of the system toolbar (Figure 35).

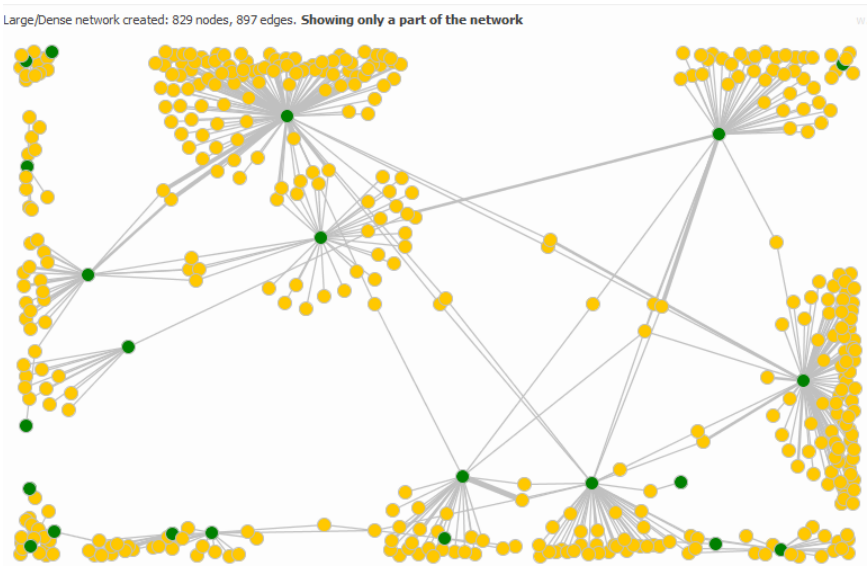



**Figure 45:** When a large network is created, the node-link visualization will sample and show a subnetwork

When slice 'n dice dimensions are specified, Ploceus shows a grid containing multiple small networks in the form of node-link visualizations only (e.g. Figure 46). This matrix of networks supports the brushing interaction technique, so that highlighting an entity in one network will highlight the same entity in other networks too.

If the dimension used contains large number of categorical values, the large number of subnetworks can lead to usability and performance problems. This is one design issue that we would like to investigate in future work.

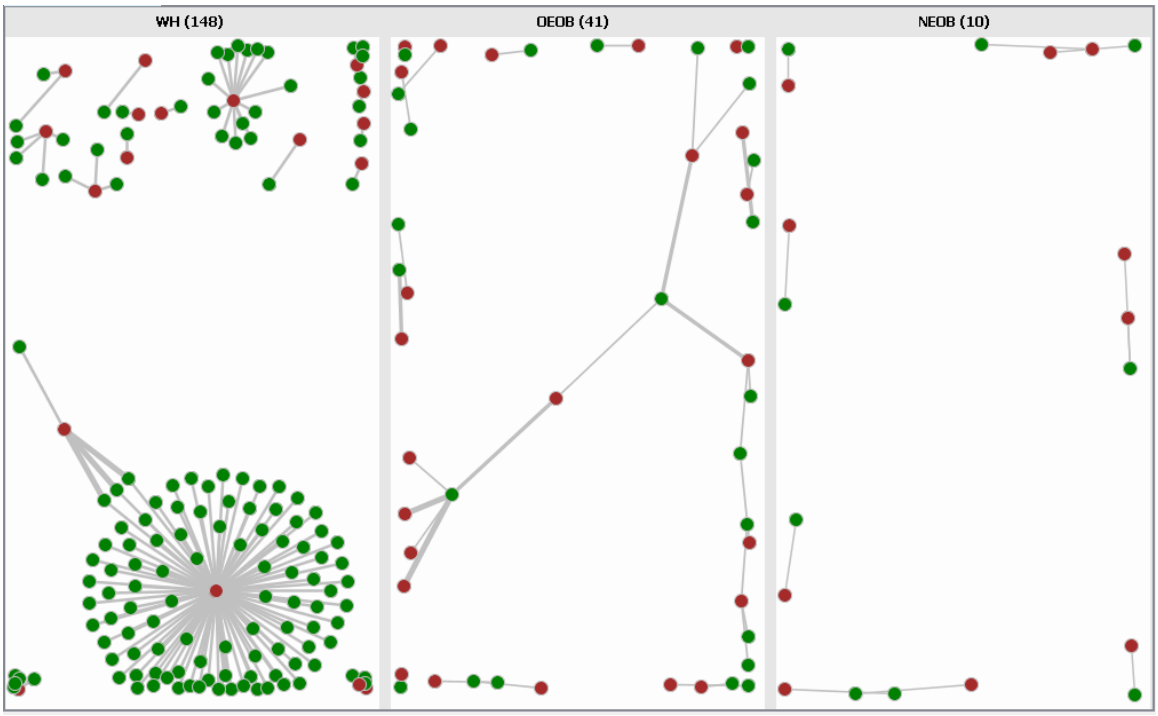



**Figure 46:** The Ploceus system interface showing a network of visitors and visitees to the White House, broken down by the locations of visits: White House, Old Executive Office Building, and New Executive Office Building

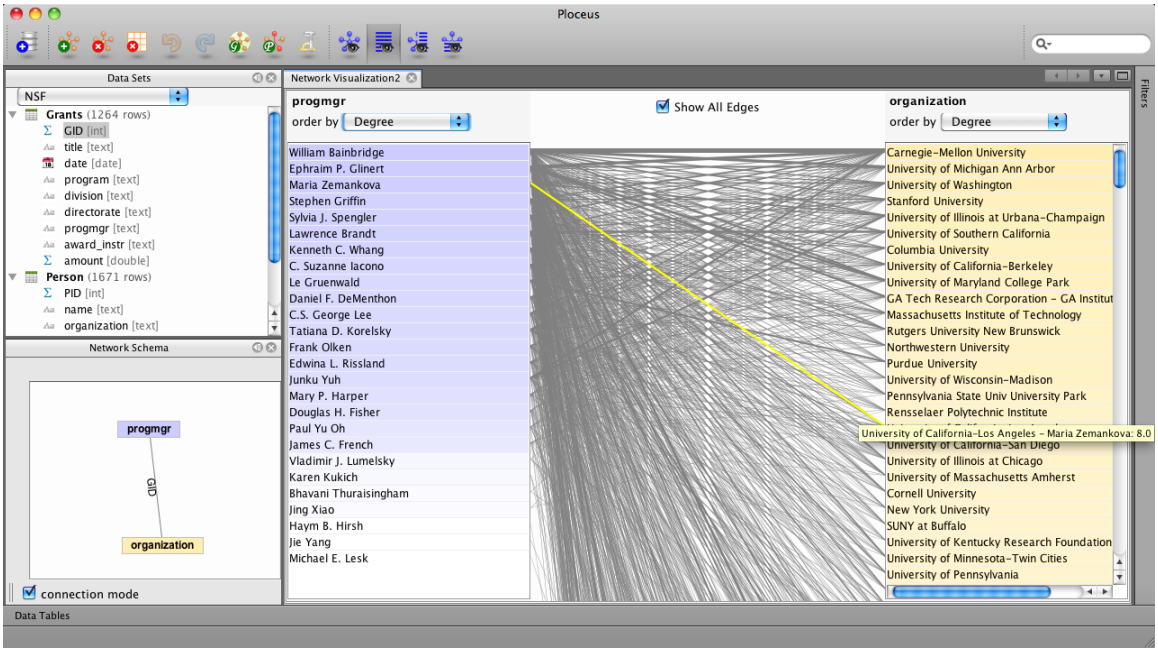



**Figure 47:** The Ploceus system interface showing a list representation of the same network presented in Figure 35

### 5.3.2 List-based Representation and Statistical Metrics

Node-link visualizations usually depict visual clusters and outliers effectively, but may not be suitable for showing the order and rankings among individual nodes. Ploceus provides easy switch to a list-based view where different types of nodes are displayed in lists (Figure 47), similar to the List View in Jigsaw [82]. When a network contains two types of nodes, the list view makes it clear that it is essentially a bipartite graph. In such list-based representations, the nodes are sorted alphabetically, or by metrics such as degree centrality, betweenness centrality and closeness centrality. According to the metric value, each node is assigned an alpha transparency value, making it easy to visually compare the nodes' values of the currently chosen metric. Unlike the node-link visualization, the layout algorithm used in the list-based visualization is lower in terms of computational complexity, so the lists show the entire network even when the size of the network exceeds the threshold.

## 5.4 Visual Encoding

The direct manipulation interface supports the articulation of operations that determine the constituent data of desired networks. Subsequently, it is important to visualize these networks appropriately. Commonly used visual variables to encode data dimensions are color, size and spatial positions. In particular, spatial position encoding, or graph layout, plays an important role in showing prominent visual structures such as clusters and outliers.

The node-link representation included in Ploceus supports five layout algorithms: Fruchterman-Reingold force-directed layout [39], circular layout, spring layout, the Kamada-Kawai algorithm [55] and Meyer's self-organizing layout [72]. The effectiveness of these layout algorithms often depends on the specific properties of the network being visualized, and analysts can experiment with different algorithms through a combo box.

In addition to providing mechanisms for spatial position encoding, Ploceus intelligently infers the appropriate visual encodings for the nodes based on the type of the underlying table dimensions. Studies have examined how people perform in perceptual tasks in terms of accuracy when different information types (quantitative, ordinal, nominal) are represented using different visual variables (e.g. area, color, density) [27, 68]. Researchers have explored the issue of automatic graphic encoding [68, 69] by incorporating these established design principles into system logic for effective visualization.

Ploceus draws from these research findings to apply effective graphic presentations in the visualization of networks. As noted in the previous section, when analysts create a new type of node, Ploceus automatically assigns a new color to that type. In addition, when analysts designate table column(s) as attributes of nodes, Ploceus analyzes the type of the column(s) and visualizes the attribute values accordingly. If analysts assign a quantitative column as node attribute, Ploceus will sum up the

quantitative values and represent it using node size. For example, after adding an attribute Size to the people nodes constructed from LName, FName in Table 1, the node "Smith, John" will have a value of 381 for the Size attribute. Figure 48 shows the resulting visualization based on a larger dataset on the White House visitor information. When analysts designate a date column as a node attribute, Ploceus treats dates as quantitative values and thus represents them using node size.

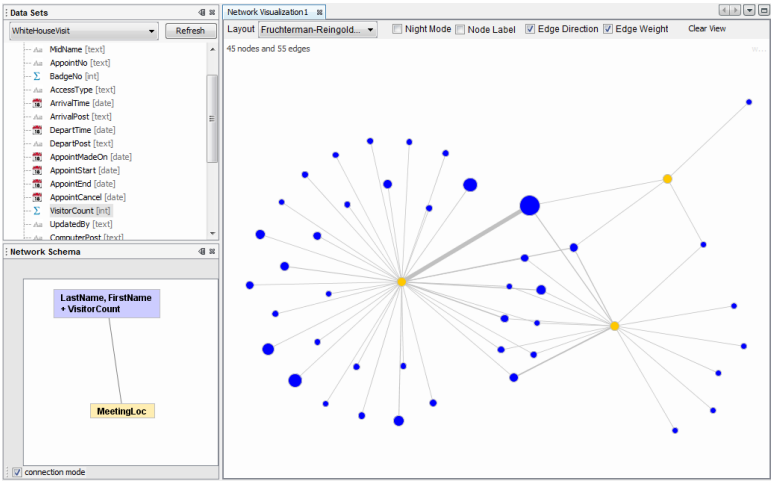



**Figure 48:** Ploceus visualizes the attribute VisitorCount of the visitor nodes constructed from LastName, FirstName as node size

Encoding a categorical node attribute is a design decision requiring more consideration. It is arguably best practice to use visually distinct colors to represent a categorical variable [94]. Alternative ways of encoding categories are to use texture or shape [68]. In any case, when there are many unique categorical values, it is difficult to define enough visually distinct representations.

In Ploceus, we use color to represent the type of node as discussed in Section 5.2. This initial decision implies that we may have to use shapes to represent categorical node attribute values. The number of visually distinct shapes, however, is limited compared to the available choice of distinct colors [93]. Assuming that analysts usually will create relatively few node types, we experimented with using shape to encode node type and using color to encode the categorical attribute values instead.

Figures 49 and 50 show visualizations generated with this approach. Figure 49 shows a network of the White House visitors and visitees, represented by circles and diamonds respectively. Ploceus assigns a default color to all the nodes. Figure 50 shows the resulting network after we add an attribute denoting the meeting location to the visitors, which is represented using color.

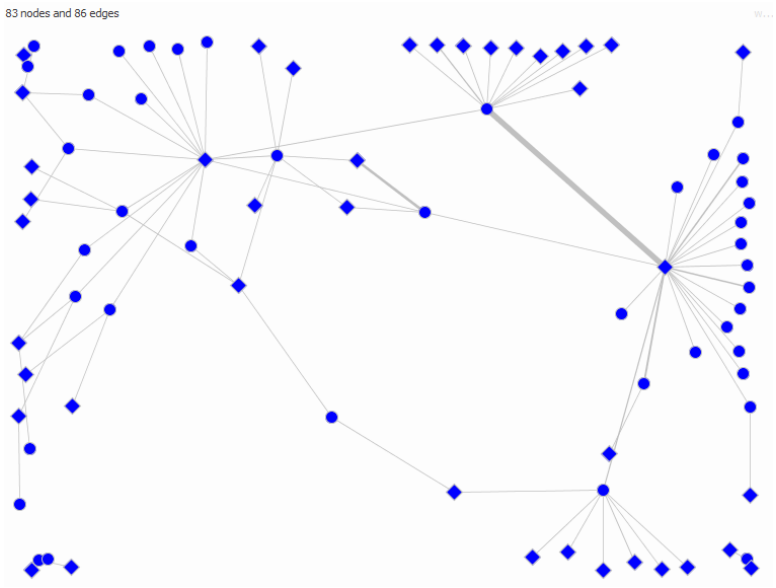



**Figure 49:** A network of the White House visitors and visitees, represented by circles and diamonds respectively

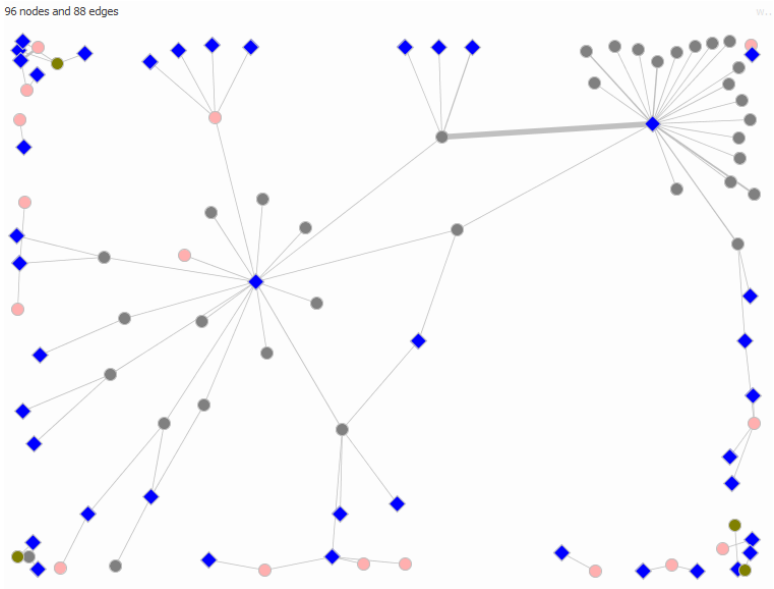



**Figure 50:** Adding "meeting location" as an attribute to the visitor nodes and representing the attribute using color

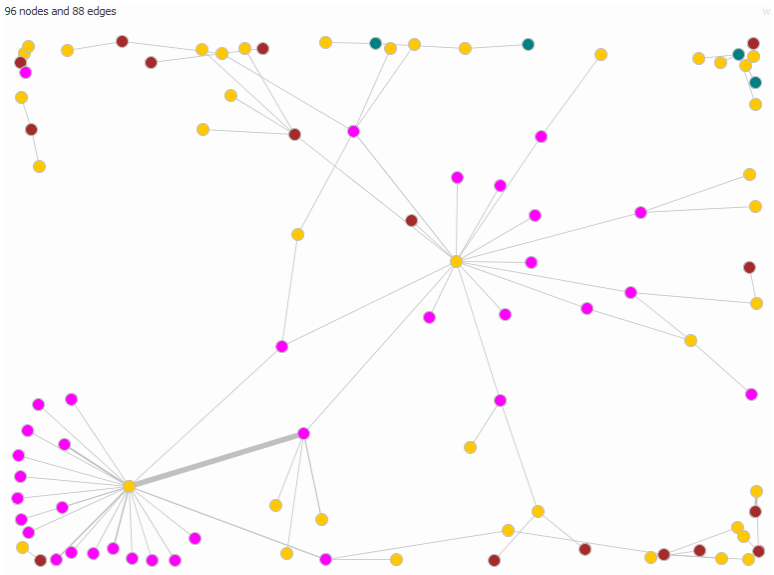



**Figure 51:** Ploceus visualizes the attribute MettingLoc of the visitor nodes constructed from LName, FName as node color

Informal feedback gathered from visualization experts on this approach was not positive, however. They strongly preferred encoding node type as color instead of shape and suggested that it was potentially confusing to interpret Figure 50. Since

node type can be considered as a node attribute too and is automatically generated, it is more essential than an optional node attribute defined by users. We thus decided to treat node type as the default node attribute and to continue encoding it using color. Analysts can define new attributes by dragging and dropping columns onto existing nodes, and if the new attribute is categorical, it will be color coded and replace the default color assigned to the node type. Figure 51 shows the visitor-visitee network, where visitees are in yellow and the visitor nodes are colored by the locations of their visits.

The work described in this section lays the groundwork for further investigation of a comprehensive graph visualization framework. The Polaris formalism [87] establishes an algebraic framework for table-based visualizations that provides effective mapping from data variables to visual variables. We envision that a similar framework is possible and is needed to describe the mappings between attribute-relationship graphs and various graph visualizations. Such a framework will be useful for automated generation of graph visualizations and may suggest visualization techniques that have not been explored before.

## *5.5 Visualization Management and Work Flow*

A direct consequence of providing a variety of construction and transformation operations is that it is now easy to generate a large number of distinct networks. Managing the networks thus becomes an important issue in the design of the user interface. In Ploceus, every network generated is associated with a tab. Analysts can generate new blank networks through the toolbar "New Network" button, and closing a tab deletes the network. Within each tab, analysts can switch between a node-link visualization and a list-based visualization; they can also tile these two visualizations side by side.

In the case of slicing and dicing, analysts can right click on any of the subnetwork and choose "Analyze in detail" in the pop-up menu. Ploceus will display the chosen

subnetwork in a new tab, where analysts can examine it more closely and change the representation to list-based visualization. In this newly created tab, Ploceus remembers the specific slice 'n dice dimension values associated with the subnetwork, so analysts can choose to delete the network while keeping the slice 'n dice values for further exploration of alternative networks from the same perspective. Whenever a new network is created or deleted, or an existing network is transformed, the network schema view will update accordingly to reflect the schema of the network in the currently active tab. Figure 52 shows an overview of the work flow in using Ploceus.

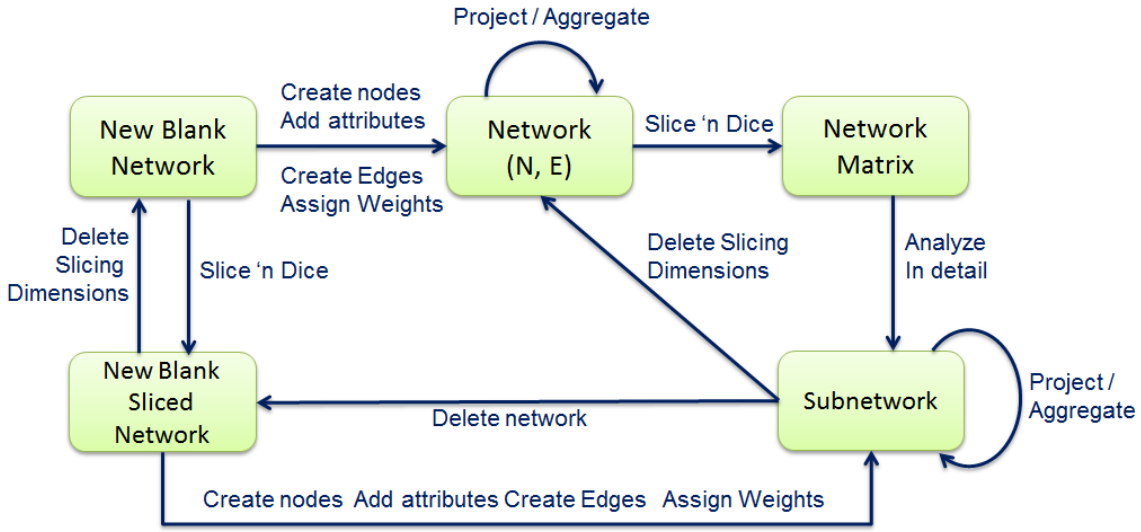



**Figure 52:** An overview of the work flow in using Ploceus. Different states of the network are shown in rectangles, and the arrows represent the user interaction to transit between the states.

## *5.6 Usage Scenario: Analyzing Cross-Institution Research Efforts*

To illustrate how to use the direct manipulation interface in conjunction with the visualization and computational capabilities provided by Ploceus for fast analytical insights, we present an example analysis in this section. For a more interactive and complete view of the analytic process, we refer readers to the accompanying video [2].

[2]The video is available online at http://vimeo.com/21773979

In this scenario we examine the research grants awarded by the NSF in the Information & Intelligent Systems division from 2000 to 2003. A subset of the data is presented in Table 3. It is a long-standing policy of NSF to encourage inter-institution research collaborations, and it would be of interest to understand the structure of collaboration networks at an organizational level. In particular, researchers from which organizations tend to collaborate with colleagues from other institutions? What factors might have influenced the collaborations?

The data set specifies an explicit 2-mode network at the actor level (PIs/co-PIs with grants). To construct a network at the organizational level, we drag and drop the organization column from the Person table and the GID column from the Grant table to the network schema view, and connect these two types of nodes. Immediately we have a network showing the connections between organizations and the grants they have received. To establish a direct linkage between organizations, we perform a projection on the GID nodes. Since we are only interested in organizations that have collaborated with at least one other organization, we filter out the organization nodes whose degree is 0. The network shown in Figure 53 results. We can see that the network is fairly well connected, with a few very small clusters detached from the main network. This indicates that the collaboration over the years is not segregated in isolated clusters, which is a positive sign.

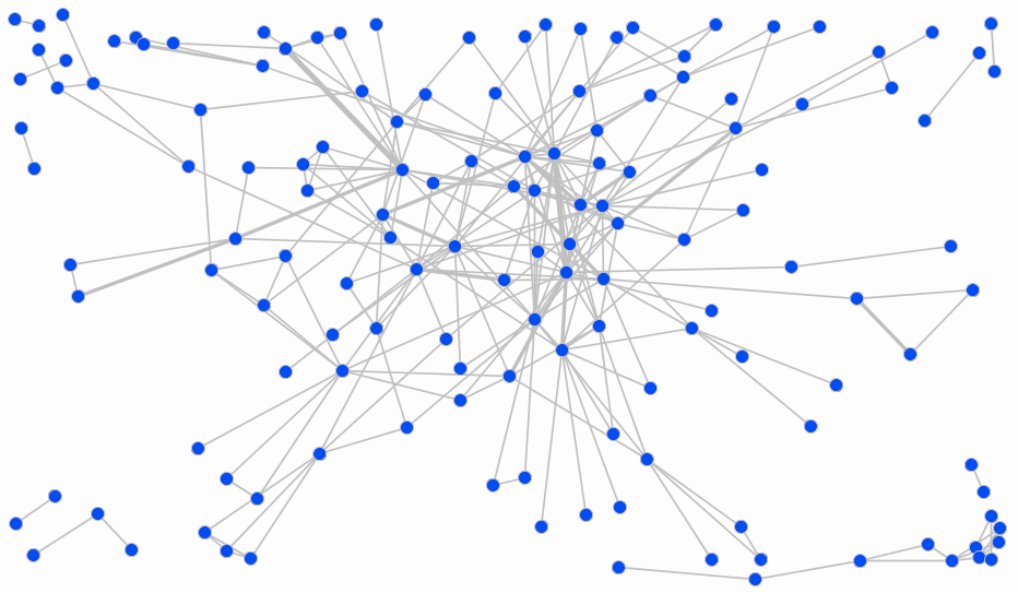


**Figure 53:** Collaboration between organizations on NSF IIS grants, 2000-2003

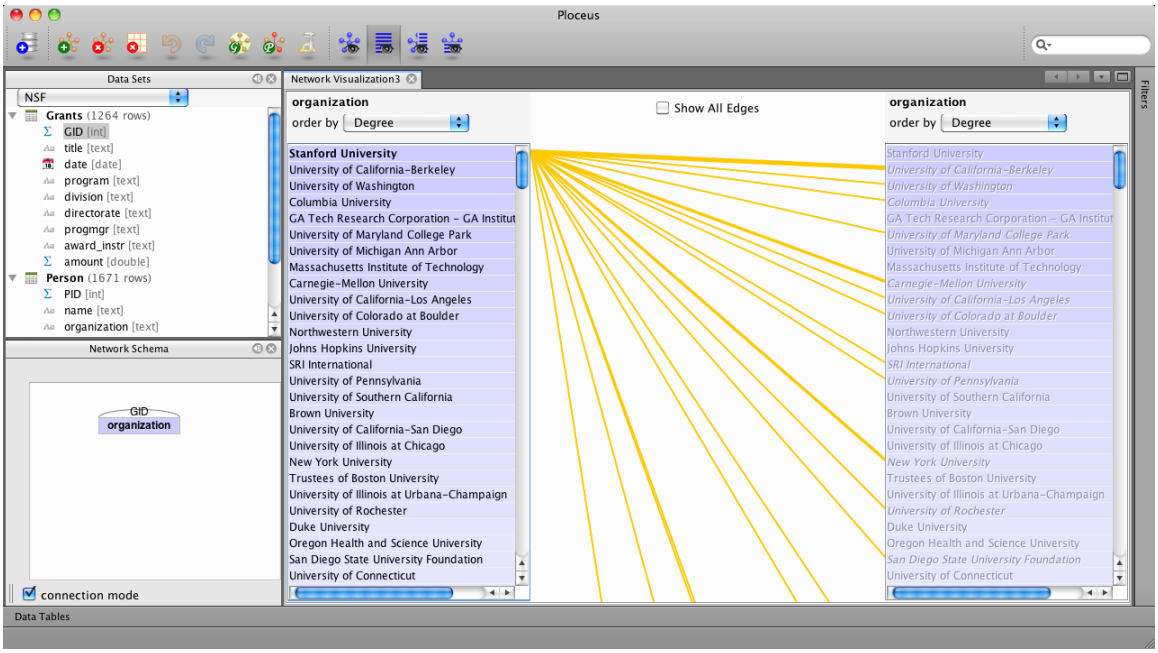



**Figure 54:** The Ploceus system interface showing a list representation of the same network presented in Figure 35

Switching to a list-based view and ranking the organizations by degrees, we see that Stanford University, University of California Berkeley, University of Washington, Columbia University and Georgia Tech are the top 5 cross-institution collaborators (Figure 54). It

is also interesting to note that Georgia Tech is the only one in the top 5 that has not collaborated with the other four organizations in the top 5.

We can continue to explore the collaboration patterns of individual organizations, but to get a more systematic view of the structure of this network first, it may make sense to slice and dice it by both the year and the amount of the award. Assuming that we have defined how the amount dimension should be aggregated into categories, this gives us the network matrix in Figure 55. The visualization here seems to conspicuously refute our intuition about the relationships between grant size and collaboration: we would expect there would be less collaboration on small grants and more on larger grants. The visualization tells us instead that medium-sized grants seem to attract the least collaborations, and this observation is fairly consistent over the four years. Considering that there were 972 small grants awarded in this period compared with 159 medium grants and 133 large grants (shown in the shelf labels), however, the sheer number of small grants might just be the main reason that increases the chance of cross-institution collaborations. Upon closer examination, we can see that grant size does also play a part in shaping the structure of collaboration networks. For small grants, two-organization collaboration is very typical, while for large grants, such collaboration patterns are much less common. In particular, there is a high level of collaboration occurring in large grants awarded in 2003.

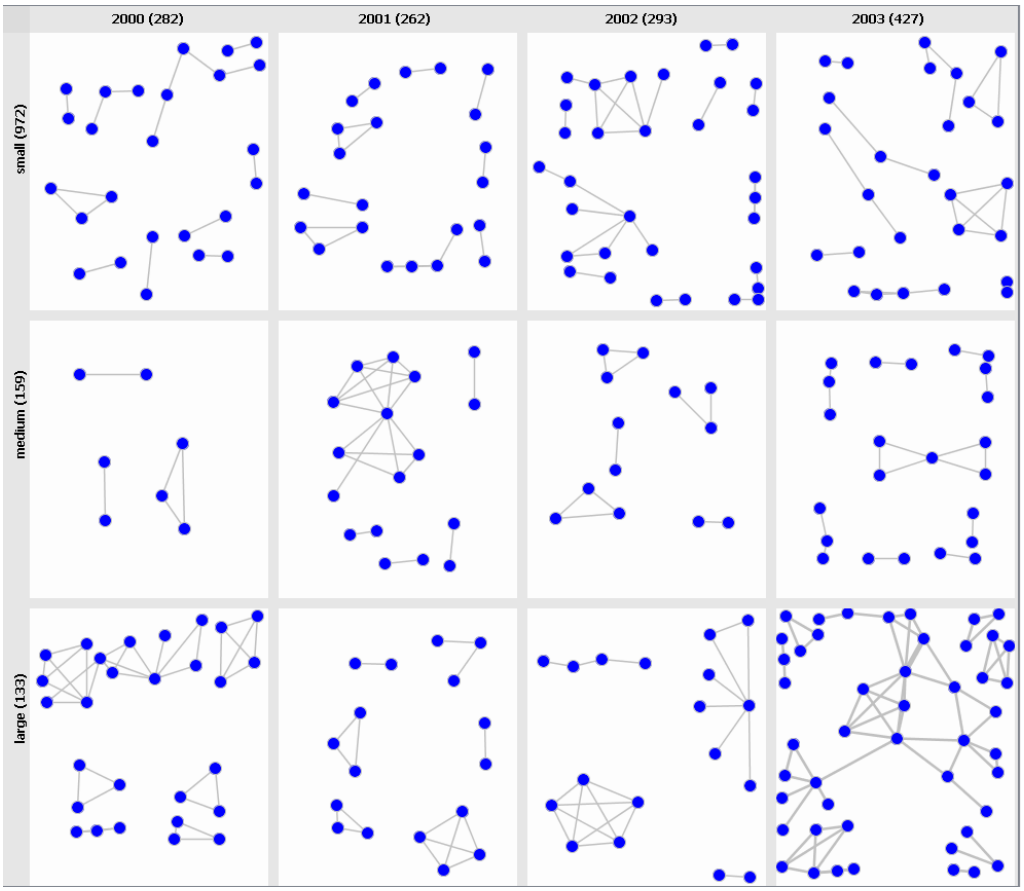



**Figure 55:** Collaboration between organizations on NSF IIS grants, broken down by year and amount

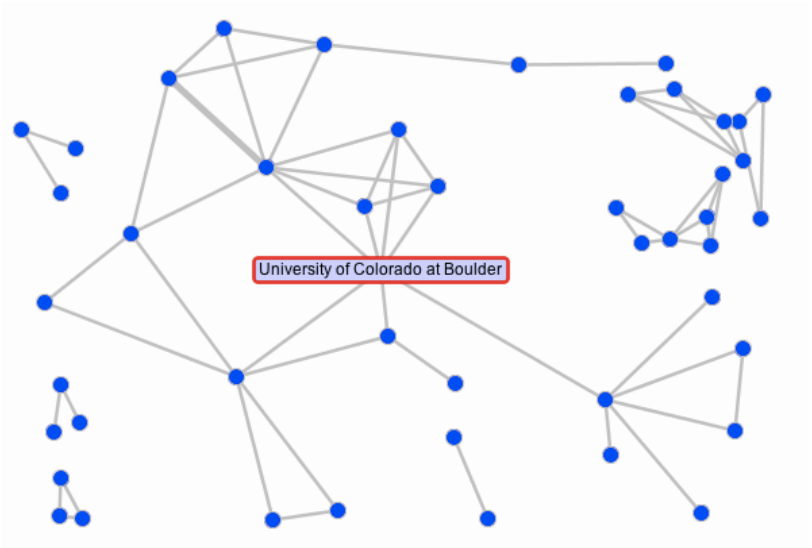



**Figure 56:** CU Boulder is an important actor in the 2003-large grant collaboration network

To investigate further, we right click in the 2003-large grant cell and choose "Analyze in detail" to open a new tab showing that subnetwork for closer analysis. We can see that University of Colorado at Boulder (CU Boulder for short) occupies an important position in this subnetwork where it connects multiple local clusters (Figure 56). This observation is confirmed after running the computational analysis, where CU Boulder has the highest betweenness centrality score, indicating that it is linking many organizations that are otherwise not linked. One reason for this is that CU Boulder has collaborated on quite a few different large grants with different organizations in 2003. To see the grants it has received as well as the collaborating institutions for each grant, we clear the current subnetwork while keeping the 2003-large grant slice specification, and construct an organization-name-title network, connecting organizations with the researchers who are connected with the grants they receive. We see the specific researchers from this school as well as the three large grants they have worked on: emotion in speech, tangible media and semantic interpretation (Figure 57).

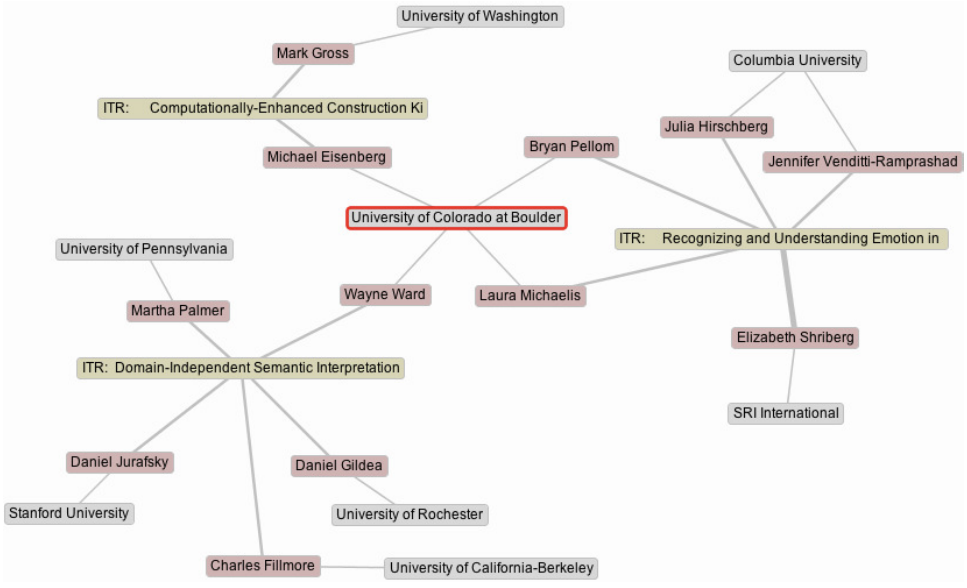



**Figure 57:** Large grants received by CU Boulder and other institutions in conjunction in 2003

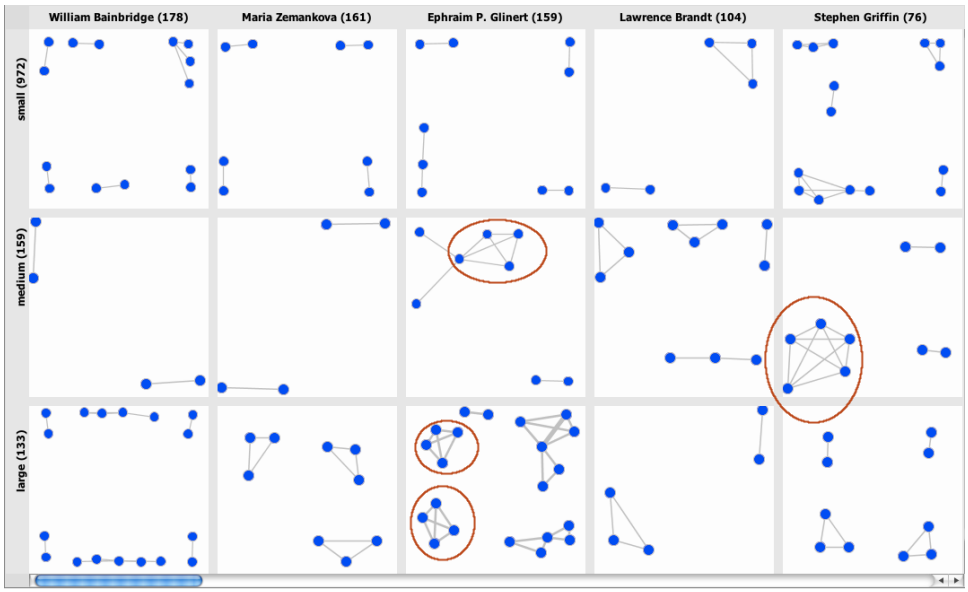



**Figure 58:** Collaboration between organizations on NSF IIS grants, broken down by program manager and amount

To look further at the role of program managers in the collaboration dynamics, we now go back to the previous tab and replace the date slices with program manager slices. Noting that William Bainbridge, Maria Zemankova, and Ephraim Glinert are the top 3 grant awarding managers, we find that a significant portion of their grants is small grants. After filtering out non-collaborating institutions, we find that grants awarded by them do not particularly show greater activities of collaboration (Figure 58). It is also obvious from the visualization that Ephraim Glinert has awarded a number of grants to groups of 4 institutions (visualized in the form of tetrahedra), and Stephen Griffin awarded one grant to a group of 5 collaborating institutions (in the form of a pentahedron). Such patterns, some of which are highlighted in the figure, are not seen in grants awarded by other program managers.

## *5.7 One-Mode Networks*

The functionalities of Ploceus and the scenarios demonstrating its utility so far have focused on modeling and visualizing multi-modal networks from tabular data. In these

tabular data sets (see Tables 1, 2 and 3), the entity relationships are *binary*, i.e. the relationships are defined between two different classes of entities. Using the projection operator provided in Ploceus, we can create one-mode networks from multi-modal networks. The system, however, did not initially provide direct support for modeling networks from tabular data that contain a *unary* relationship, defining references within one class of entities. In database entity-relationship model terminologies, such a relationship is called a reflexive or recursive relationship [34].

Table 21 shows a sample reflexive relational data set of an ego-centric social network of the user "jsmith" on Twitter. Table 21(a) records information about each Twitter user including the account (ID), the date when the user joined Twitter (Join_Date), the number of tweets designated as favorites by the user (Favorites), the number of tweets by the user (Tweets) and the self-described location (Location). Table 21(b) records the relationships between the Twitter users. We can consider this dataset to be a *one-mode directed* network. Such data are pervasive given the proliferation of social network sites and social media, and Ploceus should provide reasonable means to incorporate these data sets.

**Table 21:** Two tables describing relationships between individuals on Twitter

(a) Person

| ID | Join_Date | Favorites | Tweets | Location |
|---|---|---|---|---|
| jsmith | 2008-02-22 | 2 | 24 | Silicon Valley |
| fwong | 2008-04-04 | 20 | 231 | West Lafayette |
| jwallace | 2009-11-18 | 6 | 120 | Finland |
| ajabbar | 2010-06-25 | 30 | 15 | Paris, France |
| suzuki | 2009-05-28 | 9 | 567 | San Francisco, mostly |

(b) Relationship

| Source | Target | Relationship | Relationship_date |
|---|---|---|---|
| jsmith | ajabbar | Following | 2011-01-11 |
| fwong | jsmith | Followed | 2011-07-16 |
| jwallace | jsmith | Mention | 2011-11-01 |
| ajabbar | jsmith | Followed | 2010-09-02 |
| jsmith | fwong | Mention | 2010-02-05 |

Previous work such as PivotGraph [96] enables analysts to perform attribute-based node aggregation in a one-mode network through combo boxes. Since Ploceus provides more operations with greater flexibility, simple user interface controls may not suffice. We thus focus on extending the current interface design to support one-mode networks.

Incorporating one-mode networks into Ploceus' interface and interaction framework turns out to be relatively straightforward. Following the convention of representing one-mode networks as separate node and edge tables [79], the data management view of Ploceus shows individual columns of the node and edge tables of a one-mode network respectively (Figure 59). To provide a consistent experience in modeling both multi-modal and one-mode networks, we continue utilizing the direct manipulation paradigm. To construct a Twitter network, for example, analysts follow similar steps to those outlined in Section 5.2: they first drag and drop the ID column to the network schema view, this operation adds all the Twitter users in the data set to the network. In order to create connections, analysts must drag and drop the ID column again to the network schema view to create a dummy node. Ploceus recognizes that these two ID nodes in the schema view come from the same table column, thus treating them as the same type and assigning the same color. Finally, analysts click on one of the ID nodes in the schema view, drag and release the mouse button on the other ID node to create edges. Since this is a directed network, Ploceus supports creating directed edges when analysts hold down the "ctrl" key while using the mouse to connect nodes. Ploceus infers the condition to join the node and edge tables and creates connections accordingly.

Potentially there is an alternative way to model the same network: instead of adding ID twice to the network schema view, analysts can drag and drop the Source and Target columns to the schema view respectively, and connect these two columns. Since Source and Target are distinct columns, Ploceus will treat the nodes created from

these two columns as having different types, which is counter-intuitive. Furthermore, the source and target columns in edge tables are often numerical identifiers that refer to individual entities in node tables. Node labels created from these columns therefore are not intelligible to analysts.

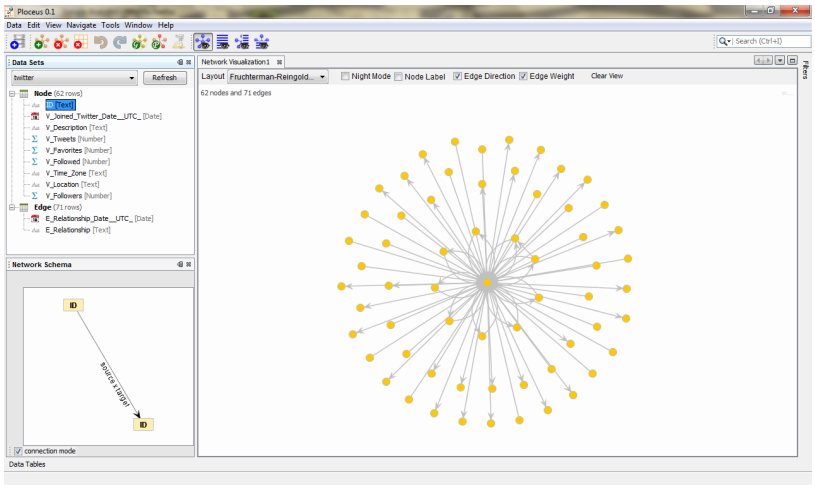



**Figure 59:** Using Ploceus to construct a one-mode directed Twitter network

Considering these factors, we make the following two design decisions. First, if analysts provide the one-mode data by importing files formatted for graph data (e.g. GraphML [2] and Pajek [5]), Ploceus parses the data and populate the node and edge tables. During this importing process, Ploceus marks the Source and Target columns in the edge table as hidden and these two columns are absent in the data management view to prevent analysts from this kind of modeling strategy. Secondly, if analysts provide the one-mode data by pointing Ploceus to an existing relational database comprising multiple tables, it is a much more difficult inferencing problem to identify the Source and Target columns. In this case, Ploceus allows analysts to aggregate two or more different types of nodes under a self-defined node type. Similar to the default aggregation discussed in Section 5.2, this "aggregate type" operation merges nodes if they share identical labels and attribute values even when these nodes are of different types.

Figure 59 shows a resulting visualization of one of the authors' own Twitter network. Ploceus represents the direction of the edges using arrows. If two nodes are connected to each other in both directions, the two edges will overlap and potentially cause confusion. Ploceus thus renders these kind of edges as quadratic Bézier parametric curves, so that bi-directional edges do not overlap and form a distinctive visual pattern (Figure 59).

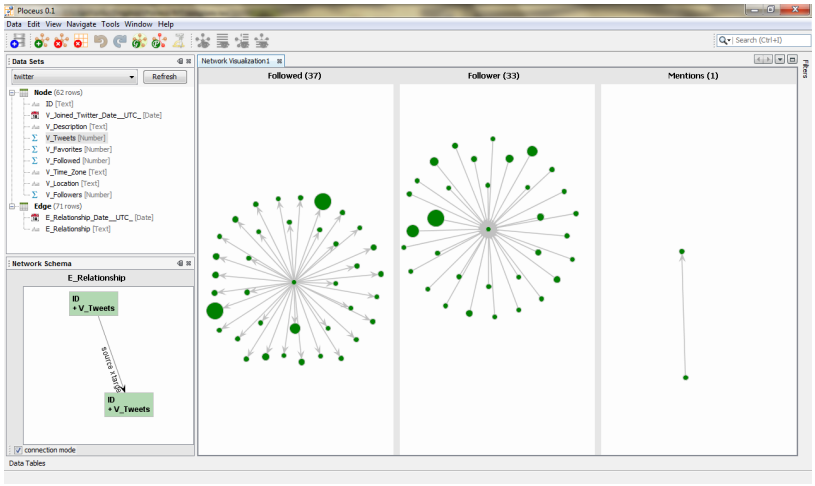



**Figure 60:** Slicing 'n dicing the twitter network using the Relationship dimension and encoding node size as the number of tweets

All the network operations such as adding attributes and slicing 'n dicing still apply for one-mode directed networks, too. Figure 60 shows three ego-centric subnetworks generated by slicing 'n dicing the network in Figure 59 using the (Relationship) dimension in the edge table: a "follower" network, a "following" network and a "mention" network. In addition, the people nodes have an attribute "Tweets", representing the number of tweets by each user, encoded as node size.

## *5.8 Implementation Issues*

### 5.8.1 Architecture Overview

Ploceus is built entirely in Java on the NetBeans Rich Client Platform [11]. It utilizes two major toolkits and libraries: H2 [10] as the underlying database for relational algebraic queries, and JUNG [73] as the graph visualization and computational metrics

library. The direct interface in Ploceus tightly integrates with a data transformation component, as indicated in the system design in Figure 34. The data transformation is based upon the formal framework described in Chapter 4. In this section, we give an overview of the system architecture. Next, we discuss implementation considerations to make data query more efficient.

A major advantage of the NetBeans Rich Client Platform is that applications based on this platform are composed of modules, which are relatively independent units of the applications that can function on their own. For example, Ploceus is implemented as a suite of the following modules:

- PloceusCore: This module defines basic data structures and all other modules depend on it. In PloceusCore, we declare four major Java packages: `ploceus.data`, `ploceus.graph`, `ploceus.io` and `ploceus.event`. `ploceus.data` defines data structures representing data set, data table, table column and user-defined aggregation bins. `ploceus.graph` contains data structures representing user-defined node and edge specifications, various specifications on user-initiated operations as outlined in the theoretical framework, node and edge instances and graphs. `ploceus.io` is responsible for data input/output, and contains classes that perform SQL queries, translate results into graph data structures, and import raw data files. Finally, `ploceus.event` declares internal system events and central controllers that are responsible for communication between different views of Ploceus. This module serves the function of "model" in the "Model-View-Controller" architectural design pattern [60].

- DateManagement: This module implements the Data Management View in Ploceus. It has a dependency on the PloceusCore module and is not aware of the existence of any other modules. The view is represented by the `DataManagementTopComponent` class, which implements the `PloceusEventListener` interface

as declared in the `ploceus.io` package. The `PloceusEventListener` interface defines an `eventReceived` method, and different implementations handle system internal events appropriately. The `DataManagementTopComponent`, for example, only listens to the `DataSourceAdded` and `DataSourceRemoved` events, and updates the data source combo box whenever a data source is added or removed.

- DataTableViewer: This module implements the Data Table Viewer in Ploceus. It has a dependency on the PloceusCore module and is not aware of the existence of any other modules. The view is represented by the `DataTableViewerTopComponent`. class, which implements the `PloceusEventListener` interface. `DataTableViewer-TopComponent` listens to the `LoadDataSourceEvent`, which is fired from the `DataManagementTopComponent`.

- GraphVisEngine: This module implements the major graphical interfaces in Ploceus. It contains five major Java packages: `ploceus.actions`, `ploceus.ctrl`, `ploceus.list`, `ploceus.nodelink` and `ploceus.schema`. `ploceus.actions` declares classes that represent user actions such as AddNewNetwork, DeleteNetwork, ProjectNetwork and Slice-Dice, each action is associated with a presenter in the global menu bar as well as the toolbar. `ploceus.ctrl` contains classes that implement the filtering panel in Ploceus. `ploceus.list` and `ploceus.nodelink` contain implementations of the list-based representation and the node-link representation. Finally `ploceus.schema` implements the Network Schema View.

### 5.8.2 Incremental vs. Holistic Approaches to Implementation

In the theoretical framework, some operations such as projection manipulate an existing network, and the transformation process does not involve the retrieval of new

data and hence is independent from the input data tables. Some other operations such as creating nodes and adding attributes instead interact heavily with the input data. As we try to seamlessly couple network modeling with visual feedback in the design of Ploceus, there are some efficiency and interactivity concerns in the actual implementation of the formal framework. In this section we discuss how to address these concerns through careful choices of SQL query generation mechanisms.

According to the interface design, the construction of a network is incremental: each user action results in an update of the visualization. For example, the process of constructing a LastNm, FirstNm – Location network from Table 1 consists of three steps: create LastNm, FirstNm nodes, create Location nodes, and create connections between LastNm, FirstNm and Location nodes (Figure 61).

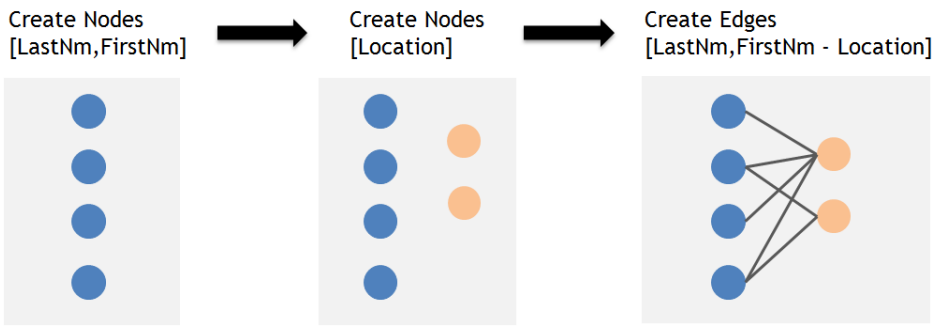



**Figure 61:** Three steps to construct a network of the White House visitors and locations in Ploceus

There are two possible ways to implement this design. The first way, characterized as ***incremental***, is to update the data structure underlying the changing visualization. The retrieval of LastNm, FirstNm nodes is accomplished through the following SQL statement:

```
SELECT ID, concat(LastNm, FirstNm)
FROM WhiteHouseVisits
```

From the result set, `concat(LastNm, FirstNm)` are used to construct node labels, and since we are assuming each tuple has a unique identifier in our framework, we

use that unique identifier (`ID`) to construct node locales. In the second step, we use similar SQL statements to retrieve and construct Location nodes. In the third step, we create edges by analyzing the intersections of the nodes' locales. Throughout this process, the data structure representing the network is updated from the SQL query results.

The second way to implement the design, characterized as ***holistic***, is to treat the visualizations in each step as independent from one another, and hence we recreate a completely new graph data structure for each step from ground up. In the first step, we use the exact same query shown above to construct LastNm, FirstNm nodes. In the second and third step, we use the same following SQL query to construct the graphs. The difference is that in the third step, we create edges from the result set while in the second step we do not.

```
SELECT ID, concat(LastNm, FirstNm), Loc
FROM WhiteHouseVisits
```

The holistic approach seems inefficient and the overlapping result sets delivered by queries in the second and third steps seem redundant. There are, however, situations where such approach to treat different stages of construction as independent makes sense. One example is the "adding attributes" operation. If we have a set of existing LastNm, FirstNm nodes, using the *incremental* approach to add attributes Type to LastNm, FirstNm nodes implies Algorithm 1.

The overall complexity of this algorithm is $O(N^2)$, $N$ being the number of tuples in the $WhiteHouseVisits$ relation. We need to do nested iterations because adding attributes does not merely append attribute values to existing nodes. Per our design highlighted in Section 5.2, Ploceus only allows a node to have one value for any particular attribute. We thus need to break apart an existing node if there are multiple values for the attributes.

**Algorithm 1** Adding attribute Type to LastNm, FirstNm nodes using the incremental approach

```
Execute SQL query: SELECT ID,Type From WhiteHouseVisits
Get result set RS
for each tuple t ∈ RS do
  for each LastNm, FirstNm node n do
    if n.locale contains t.ID then
      if n.Type ≠ null || n.Type == t.Type then
        n.Type = t.Type
      else
        new node n' = duplicate(n)
        n'.locale.add(t.ID)
        n.locale.remove(t.ID)
        n'.Type = t.Type
      end if
    end if
  end for
end for
```

The holistic approach, on the other hand, is more efficient. Algorithm 2 shows the creation of LastNm, FirstNm nodes with an attribute Type:

**Algorithm 2** Creating LastNm, FirstNm nodes with an attribute Type using the holistic approach

```
Execute SQL query:
SELECT ID, concat(LastNm,FirstNm) as Nm, Type From WhiteHouseVisits
Get result set RS
Define graph G
for each tuple t ∈ RS do
  new node n = t.Nm
  n.Type = t.Type
  if G.contains(n) then
    G.addNode(n)
  end if
  n.locale.add(t.ID)
end for
```

The complexity of this algorithm is $O(N)$, assuming that we have a hashing mechanism to quickly find a node in the graph. Presumably we can also implement a hashing mechanism to look up a node by locale or primary key of a relation for the incremental approach, thus reducing its complexity. Our experience of implementing

this approach, however, quickly results in a high overhead cost of maintaining the hash table.

We eventually chose a mixed approach to the implementation of Ploceus. For operations such as creating nodes, an incremental approach is appropriate. For some other operations, a holistic approach makes sense to improve the interactivity of Ploceus. Table 22 summarizes our eventual implementation choices as well as the run-time complexity for each operation. The complexity for the aggregation operation depends on the type of aggregation. For example, the complexity for proximity-grouping is $O(n^2)$ because we need to do pair-wise comparison; on the other hand, the complexity for binning is $O(n)$.

**Table 22:** Implementation choices for each operation in Ploceus

| **Operation** | **Single Table** | **Multiple Tables** |
|---|---|---|
| Create Nodes | Incremental ($O(n)$) | Incremental ($O(n)$) |
| Add Attributes | Holistic ($O(n)$) | Holistic ($O(n)$) |
| Create Edges | Incremental ($O(n^2)$) | Holistic ($O(n)$) |
| Assign Weights | Incremental ($O(n)$) | Holistic ($O(n)$) |
| Aggregate | Incremental (depends) | Incremental (depends) |
| Project | Incremental ($O(n^2)$) | Incremental ($O(n^2)$) |
| Slice 'n Dice | Holistic ($O(n^2)$) | Holistic ($O(n^2)$) |

Through these implementation considerations, all the operations supported by Ploceus are performed in real time: simple operations such as adding nodes and creating connections are scalable for up to tens of thousands of rows without significant delay. More complex operations such as projection and statistical metrics computation are more computationally expensive and the performance can be affected with large data sets. Every subnetwork in slicing and dicing is created through a separate thread, and the performance bottleneck is at the concurrent handling of SQL queries by the underlying H2 database.

Further improvement in terms of scalability and performance will likely comes from the building of a data warehousing model for efficient network construction and transformation. In building a data warehousing model, a primary concern is

to investigate the feasibility of full materialization of all possible networks, and the issue of balancing query performance and resource constraints in the case of partial materialization.

## *5.9 User Evaluation*

To understand the implications of the algebraic framework and the interface design, we conducted an informal evaluation of the usability and usefulness of Ploceus. We recruited four participants, all of them are graduate students knowledgeable of basic concepts in information visualization, graph representation, and visual analysis. All of them had never seen or used Ploceus. The goal of evaluation is to identify qualitative insights about the way people think about network visualization construction and how well Ploceus supports network modeling.

We gave a brief introduction on Ploceus, demonstrated the main functionalities of the system and showed the participants how to construct different networks using the White House Visitor data set (Table 1 shows sample rows). We then asked them to create visualizations and answer the following questions on the National Science Foundation data set (Table 3 shows sample rows):

- Can you create a visualization showing the collaboration pattern between organizations on research grants?
- Which organization(s) tend to collaborate the most?
- In which year(s) do we see the most cross-organization collaboration?

We asked the participants to think aloud, observed their interaction with the system, recorded their interaction history as hand-written notes, and sought their impressions and comments on the system. We wanted to gain an understanding of how difficult the system is to learn and use, if there are any aspects that are particularly problematic, and how we might be able to address these difficulties.

To answer the given questions, the participants need to create visualizations in multiple steps as outlined in the scenario in Section 5.6: first, add the grant IDs and the organizations as nodes; then connect these two types of nodes; perform a projection on the grant IDs so that the organizations are connected directly to each other via common grant IDs, and finally slice and dice the network using the date dimension to break down the network by year.

We expected that it would be non-trivial for first-time users to figure out the exact sequences of operations to construct the desired visualization. It turned out that two of the participants did not experience any difficulty while the other two did experiment a few times before discovering the correct strategy.

### 5.9.1 Identifying Leverage Points in Visualization Construction

As the designer of Ploceus, we perform a preliminary analysis of expected leverage points in the process of visualization construction. Specifically, we anticipate that using Ploceus to construct network visualizations involves the following aspects of knowledge or skills:

1. **Interpreting questions or tasks**: Users need to be able to understand what kind of insight is being inquired.
2. **Simulating appropriate visualizations**: Users should be able to mentally simulate and visualize possible visualizations that can provide the desired insight.
3. **Conceiving construction strategy**: Given a desired visualization, users need to identify appropriate operations in our framework and how these operations should be combined sequentially to create the visualization.
4. **Executing construction sequence**: To be able to execute the conceived construction strategy, users have to know how each operation can be performed through the Ploceus interface.

5. **Interpreting Visualizations**: Finally, assuming that the construction is successful, users must be able to accurately interpret the visualizations generated and provide answers to the given questions.

By examining the interaction history of the two participants who struggled with finding the correct operations, we discovered evidence pointing to potential difficulties at all five leverage points discussed above. We organize these pieces of evidence and identify two major gaps that prevented successful visualization creation: visualization gap and construction gap.

### Visualization Gap

One participant (P3) appeared to have understood the features of Ploceus after the demo session, and quickly constructed a visualization after we gave her the questions. Interestingly she created a network connecting researchers with organizations and then did a projection on the organizations. When we asked her to describe what the visualization was showing, she realized that she was not creating a visualization that would answer the questions. She then created a different network in which researchers are connected to each other if they come from the same organizations, and again this visualization was not able to answer the given questions. She finally realized what went wrong, and commented that "I totally mis-interpreted the question". After careful consideration of the semantics of the questions, she successfully created the intended visualization. We interpret from this process that she had difficulty *interpreting what questions are asking.*

Some users showed signs of a lack of ability to *mentally visualize the representation* that would answer a given question. One participant (P4) successfully created a one-mode network of collaborating organizations. To answer the question "In which year(s) do we see the most cross-organization collaboration?", he added the Date

column values as nodes to the visualization, and connected the dates with the organizations. He tried hard to answer the question by examining the visualization, but did not have any viable lead.

### Construction Gap

Even if users know clearly what visualization they would like to see, there are still potential difficulties to conceive the appropriate steps to construct the visualization. Previous studies have shown that visualization construction is a major hurdle for novice users [43]. In observing the participants, we get similar impressions. P4 wanted to create a one-mode network by projection, but he forgot to create the connections between the two classes of nodes first, and the projection could not be performed. In this case, he could not *conceive a proper sequence of the operations* to construct the visualization.

Most of the users did not have any difficulties in translating an intention of performing an operation to an actual action. That is, they were able to pinpoint the user interface component designed to support a specific operation. One participant (P4), however, could not figure out how to perform slicing 'n dicing even though we demonstrated this functionality for him. He commented, "Although I've seen it, I just completely forgot about it". We acknowledge that the demonstration session was relatively short, and viewing a demonstration is very different from performing the same action oneself. The participant was confident that he would perform much better if he had experimented with Ploceus for a longer period of time, however.

### Interpreting Visualizations

All users were familiar with node-link diagrams and had no difficulty in reading the visualizations generated. The major difficulty surfaced when the size of the generated network became too large. Specifically, The layout algorithms included in Ploceus

can be improved. The force-directed layout, for example, takes some time to settle down for a network with more than 500 nodes, and it does not show clusters inherent in a network clearly. Disconnected subnetworks tend to be pushed to the boundaries of the visualization, thus making the graph less readable.

Visualization of large graphs remains a challenging issue, as we are constrained by the limited number of pixels on the screen. Interaction with subgraphs is mainly accomplished through expanding neighbors of a focus node, and is not very efficient when we want to explore a subgraph including all the nodes that are $k$ paths away from the node of interest. Incorporating more sophisticated techniques like the Degree Of Interest function [91] is a promising direction to enhance interaction in Ploceus.

### 5.9.2 General Impression and Comments

All the participants liked Ploceus, especially the interface design. Although some participants considered constructing network visualizations non-trivial, they still agreed that the interface was consistent and the learning curve was not too high. The affordance of the network schema view was clear to all the participants, and all of them strongly favored this view.

The participants also identified features in Ploceus that they had difficulties in understanding and interpreting. More than one participant mentioned that the affordances of slicing 'n dicing shelves were not immediately clear to first-time users, but they acknowledged that after some interaction the design became to make sense for them. One participant also mentioned that the meaning of numerical value following each slicing 'n dicing value (e.g. when we slice 'n dice an organization collaboration network by year, a slice will be displayed as "2000 (282)", as shown in Figure 55, indicating the number of grants given in year 2000) was not clear.

One participant was so interested in Ploceus that he asked for some extra time after the given tasks to perform some open-ended exploration. One of the questions

he wanted to know was who got the most money from NSF. He tried to create a network connecting researchers with amounts, and then tried to order the amount nodes by value in the list view. While this is certainly one way to answer the question, a bar chart might be a more effective visual representation than a network visualization. In this regard, the user should have picked a system like Tableau [6] instead of Ploceus. This observation is consistent with what Kobsa [59] has noted in his evaluation study of early InfoVis systems: users tend to use the default system or setting given to them, and it is difficult for them to initialize a change in the mindset to explore alternative representations or system settings. One potential solution to this problem is to incorporate good visualization practice and design principles into the system intelligently. We discuss an inspired idea for future work on natural language interfaces in Section 6.2.5.

# CHAPTER VI

# CONCLUSION

## *6.1 Summary*

The goal of this dissertation is to provide a novel visual analytics methodology to explore multidimensional tabular data. In Chapter 3, we discuss various visual and computational methods for analyzing data tables. These methods, however, mostly focus on the analysis of quantitative and ordinal dimensions. A network, as both a data structure and a visual metaphor, provides a potentially interesting angle to look at tabular data analysis differently, especially when data tables contain important nominal dimensions.

Motivated by this observation, we present a formal framework in Chapter 4 with the goal of supporting systematic and flexible network modeling from tabular data. We base our framework on the relational model used extensively in database theory, and cover the mechanisms of constructing and transforming networks from both a single relation and multiple linked relations.

In Chapter 5, we demonstrate how the framework can be useful as a theoretical foundation for the design and implementation of a visual analytics system Ploceus. We discuss the interface design choices involved in building Ploceus with the goal of providing a user-centric, intuitive and seamless exploratory data analysis experience. Relating to the formal framework, we present different approaches of the system implementation. Finally, we show the potential usefulness of Ploceus through a detailed analysis scenario.

In summary, we have proposed and demonstrated a viable method for analyzing tabular data using network modeling and analysis techniques. Since tabular data such

as spreadsheets and relational databases are pervasive, we expect our framework and the Ploceus system to have wide applicability in many problem domains.

## *6.2 Future Work*

The formal framework and the Ploceus system are the first step of a grander agenda of exploring a network-oriented exploratory data analysis method of tabular data. Through our experience of designing the framework, applying it to sample datasets, and assessing the semantic and analytic power of our approach, we discover three major issues that are not solved completely by the current framework and system implementation. In addition, we discuss future directions for system evaluation and user studies.

### 6.2.1 Joining Multiple Tables

As we have seen in Section 4.2.2, the relational join is a crucial concept in constructing graphs from multiple data tables. While Ploceus uses the Dijkstra shortest-path algorithm [31] to analyze foreign-key dependencies and to infer equi-join conditions, when a database becomes more complex in terms of Entity-Relationship modeling, there might be multiple reasonable equi-join conditions.

Consider an IMDB (Internet Movie Database) dataset containing information about movies, persons working on the movies and the awards related to these movies. Figure 62 shows the ER schema, where a Movie table is connected to a Genre table through a many-to-many relationship (a movie can simultaneously be a drama, a war movie, an adventure movie, for example), the Movie table is connected to a Person table through a many-to-many Works_on relationship, and the same Movie table is connected to the same Person table through a many-to-many Oscar relationship.

Suppose we want to create a Person − Genre network. The Person nodes are constructed from the Name column in the Person table, and the Genre nodes are constructed from the Genre column in the Genre table. Now the Dijkstra shortest-path

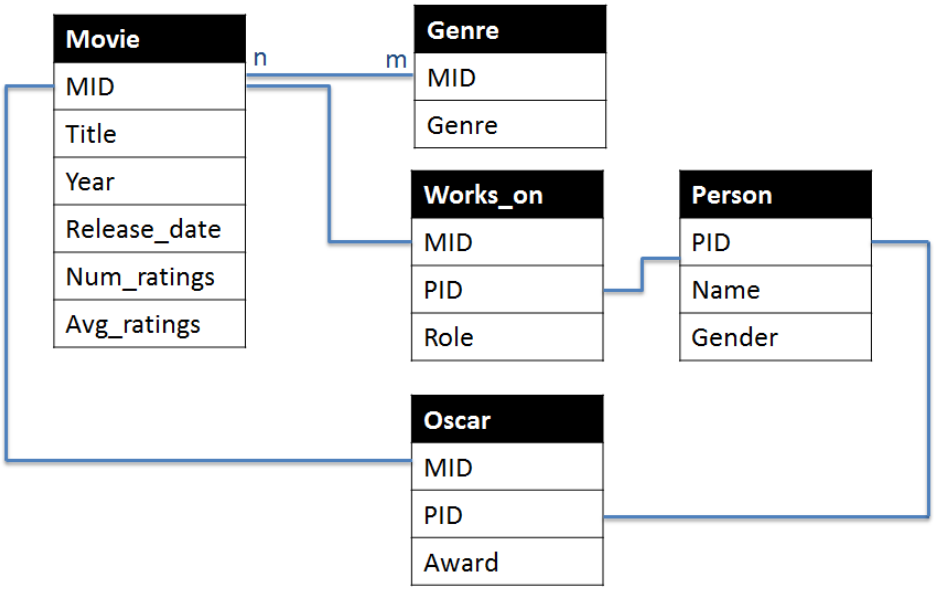



**Figure 62:** The ER schema of an IMDB (Internet Movie Database) dataset containing information about movies, persons working on the movies and the awards related to these movies

algorithm will tell us that there are two potential paths that link the Person column with the Genre column: through the Works_on table, and through the Oscar table. The resulting networks thus will have different semantics: one showing the relationship of Oscar winners and the genre of movies which win them an award; the other showing the relationship between any one involved in producing a movie and the genre of that movie.

When multiple equi-join conditions are available, it is the user's decision to choose the appropriate join condition. Currently Ploceus handles this situation through a dialog, letting users interactively add relevant tables, and connect the primary keys and foreign keys to specify the desired join condition (Figure 63). Related systems such as Tableau [6] take a similar approach. Tableau does not support interactive table joining during the process of exploration. Instead, at the stage of data import, analysts need to explicitly join tables to include all the data columns necessary for exploration. Tableau supports a richer set of join types and conditions (Figure 64).

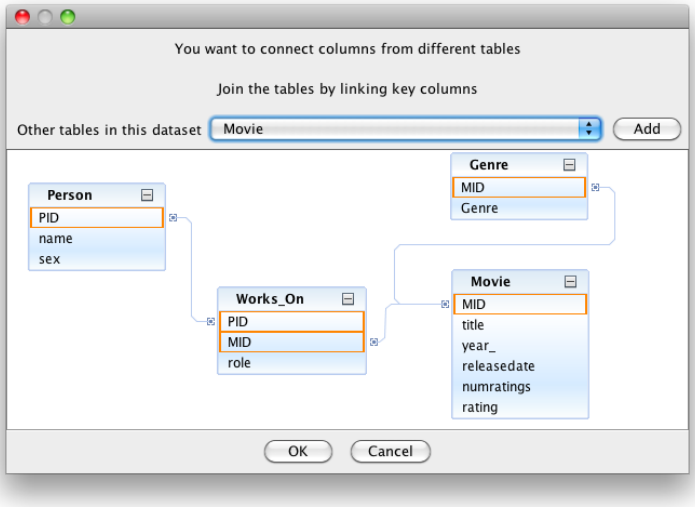



**Figure 63:** Users interactively add relevant tables and connect the primary keys and foreign keys to specify the desired join condition in Ploceus

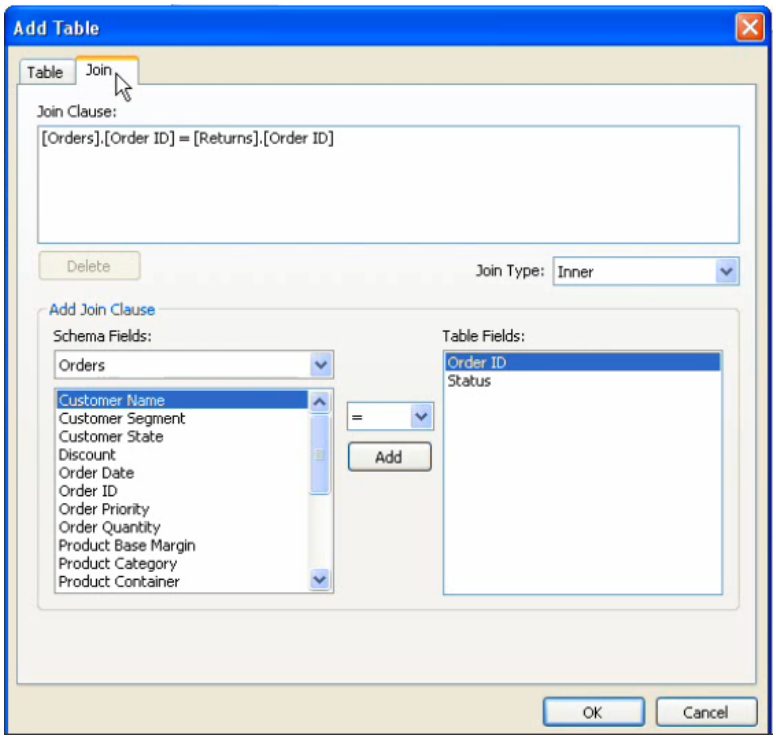



**Figure 64:** The dialog interface in Tableau to join multiple tables

Potentially we can also use more sophisticated techniques to automatically compute a number of different join conditions and to rank them by inferring analysts'

intention. The diversity of all the possible join conditions, however, can hardly be fully captured. More importantly, all these approaches do not address a fundamental issue satisfactorily: analysts must have a precise and good understanding of the concepts of relational join, primary key and foreign key. Even if they understand these concepts well, it is still non-trivial to interpret the semantics of edges constructed as a result of joining multiple tables.

Currently we do not have a satisfactory solution to this problem, and we doubt there will be one if the underlying data model is going to be relational. The recent emerging NoSQL databases [71] might provide an interesting angle to address this issue. Multiple related tables in relational databases are a direct consequence of the design choice to normalize data tables for the sake of minimizing redundant data representation and avoiding anomalies in data modification [34]. These goals are not emphasized in NoSQL databases. It would be interesting to explore if the problem of mandatory join specification can then be eliminated if we choose NoSQL data model (key-value pairs instead of data tables) as our input data model.

### 6.2.2 From Operations to Algorithms

The formal framework defines a set of basic operators for network construction, and Ploceus provides a user-centered interface of these operators. As we have discussed in Section 4.4, there may be limitations regarding the expressive power of the current set of operators. It is not clear, however, whether our formal framework can be responsible for all the possible expressive power needed.

Consider a dataset about researchers and their academic publications. A Paper table describes information about academic papers, and a Researcher table describes information about researchers. The Paper table and the Researcher table are connected through a many-to-many Author – Of relationship. In addition, a unary reflexive Citation relationship exists for the Paper table.

**Table 23:** Tables describing researchers and the papers they publish

(a) Paper

| PID | Title | Year | Venue |
|---|---|---|---|
| 1 | Jigsaw: Supporting Investigative Analysis through Interactive Visualization | 2008 | *Information Visualization* |
| 2 | Sensemaking Processes of Intelligence Analysts and Possible Leverage Points as Identified Through Cognitive Task Analysis | 2005 | International Conference on Intelligence Analysis |
| 3 | The cost structure of sensemaking | 1993 | CHI |

(b) Researcher

| RID | Name | Org |
|---|---|---|
| 1 | J. Stasko | Georgia Tech |
| 2 | C. Görg | Georgia Tech |
| 3 | Z. Liu | Georgia Tech |
| 4 | S. K. Card | PARC |
| 5 | P. Pirolli | PARC |
| 6 | M. J. Stefik | PARC |
| 7 | D. M. Russell | PARC |

(c) Author – Of

| Author | Paper |
|---|---|
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 2 |
| 5 | 2 |
| 4 | 3 |
| 5 | 3 |
| 6 | 3 |
| 7 | 3 |

(d) Citation

| Citing | Cited |
|---|---|
| 1 | 2 |
| 2 | 3 |

Suppose we want to construct a one mode directed network of researchers, who are connected by edges representing citation relationships. Our formal framework supports the construction of the two-mode network shown in Figure 65. Two types of edges exist in this graph. The edges between researchers and papers carry the meaning of authorship: a researcher and a paper are connected if the researcher is an author of that paper. The edges between papers are directed and have the meaning of citation relationship: a source paper node cites a target paper node.

The network we want to construct is shown in Figure 66. The edges between researcher nodes indicate the relationship of citation. Currently in our framework, no operation can transform the network in Figure 65 to that in Figure 66. Projection

is the closest candidate, but the required semantics is richer than that of projection. Actually, what we want to achieve can be captured by a simple algorithm shown in Algorithm 3: in the two mode graph, for each paper authored by a researcher $r$, we find all the papers cited by the paper, and create a directed edge from $r$ to every author of each of these cited papers.

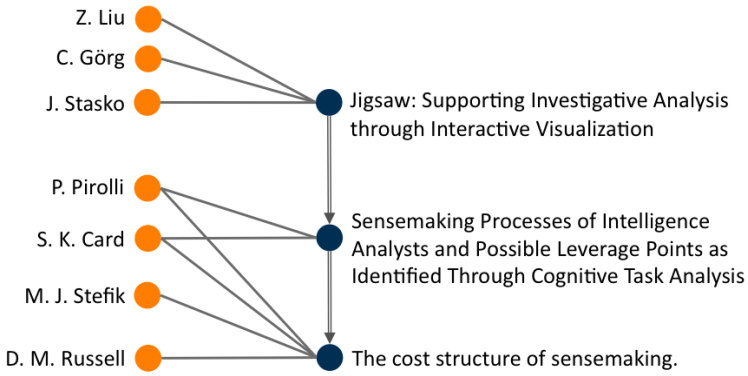



**Figure 65:** A two mode graph of researchers and papers. A researcher and a paper are connected if the researcher is an author of that paper. A source paper node cites a target paper node.

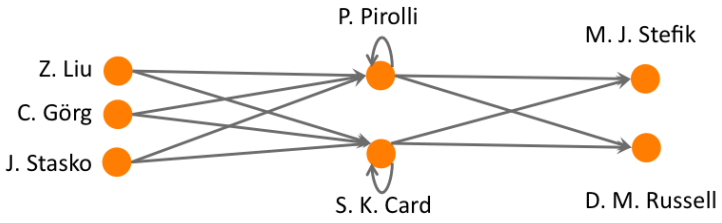



**Figure 66:** A one mode directed network of researchers. The relationship between researchers represents paper citation.

Such algorithms are more about transforming networks than about constructing networks from tabular data. Conceivably we can write a huge variety of such algorithms, taking into account the particular semantic details of the dataset being explored. It would be beneficial if our framework could provide a mechanism to

**Algorithm 3** Algorithm to transfrom the network $G$ in Figure 65 to that in Figure 66

```
for each author node n ∈ G do
  for each author node m ∈ G do
    P = {all paths from n to m}
    for each path p ∈ P do
      if p.length = 4 then
        G.addEdge(n,m)
      end if
    end for
  end for
end for
```

incorporate these diverse algorithms. We believe, however, the integration of customized algorithms should not be at a theoretical level, but may be implemented at the system design level. Generic systems such as KNIME [18] and Excel DataScope [9] offer plugin-based architectures to allow flexible implementation of customized computational techniques in the form of stand-alone modules. Since we implement Ploceus using a modular framework (the NetBeans Platform), it is possible to orient future system development towards this direction.

### 6.2.3 Formal Evaluation

Moving from research to real-world adoption, we are interested in the ecological validity and implications of our design. Through our informal evaluation, we have gained insights on the implications of our design choices. The examples and analysis scenarios also demonstrated the potential utility of our framework and the Ploceus system. These findings will need to be strengthened and vindicated through real-world use case scenarios. We would like to know if the utility of the system varies according to the semantics of the data set and problem domain, and if integrating network modeling with exploratory analysis provides enhanced analytic power, thus leading to more insights about the data. Performing Multi-dimensional In-depth Long-term Case Studies [78] is an appropriate evaluation strategy we can pursue. Case studies

provide ecologically valid evidence that can demonstrate the potential benefits and shortcomings of the system. The longitudinal nature also makes possible detailed understanding of how analysts' strategies evolve over time.

A comparative study can provide insights about the design of interface in Ploceus as compared with alternative systems with similar design goals. The Orion system [49] is a potential candidate where it is designed for the similar purpose of supporting network modeling. Their approach, however, is different from ours in that they expose more technical details to the users and put a stronger focus on automated graph edge construction using path finding algorithms. Our approach is more human-centered. For example, to create nodes from a table, users will "promote" a raw table column to become a full table in their system Orion. In Ploceus, this operation is achieved when users drag and drop table columns, and users do not have to be concerned about how the newly created nodes are represented at the data structure level. To create a one-mode network, the Orion system will find all possible linking paths between raw data tables as well as "promoted" data tables and list them for users to choose (Figure 67). In Ploceus, analysts will likely perform such construction via multiple steps using node creation, edge creation and projection operations. In doing so, we are assuming that analysts know what network they want to construct, and they retain full control of the construction process.

The differences in these design choices represent trade-offs that have implications on system usability. We can ask participants to perform the same network modeling tasks using Ploceus and Orion respectively and compare their performance. Since Orion is essentially a stand-alone modeling interface that is decoupled from network exploratory analysis, such a comparative study will also shed light on the utility, if any, of providing immediate visual feedback in the process of model-based visual analysis.

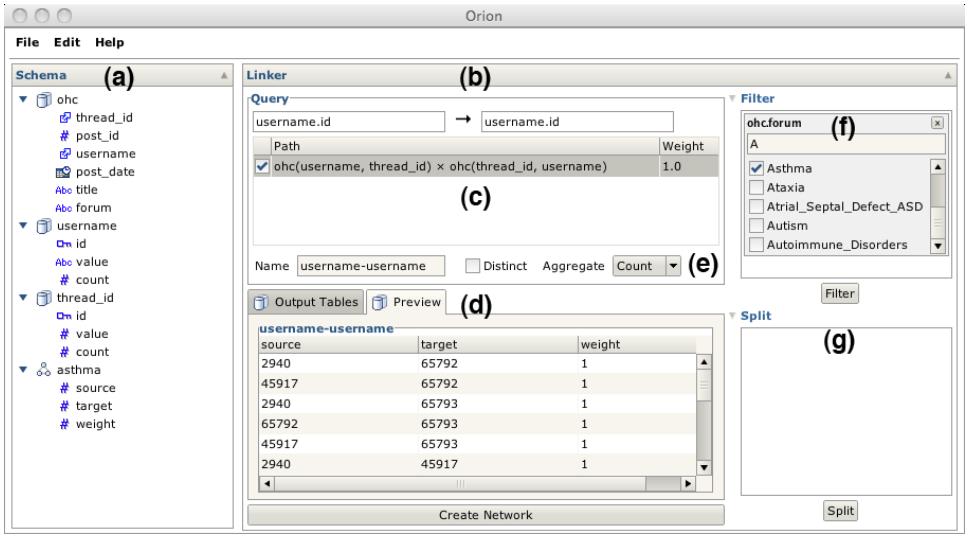



**Figure 67:** The Orion User Interface, consisting of (a) a schema viewer for manipulating data tables and (b) a linker interface for creating network models. Analysts drag-and-drop desired node types to the linker and Orion responds with (c) a table of possible linking paths. The (d) preview display shows the resulting network data.

### 6.2.4 Big/Dense Graphs

Scalability is an important issue in Information Visualization and Visual Analytics research. An implication of the network modeling power provided by Ploceus is that analysts can easily create both big graphs with thousands or even millions of nodes and dense graphs where the number of edges is close to the maximum possible number of edges. Visualizing and analyzing these graphs remain great research challenges. Due to the limited number of pixels available on computer screens, it is impossible to display all the nodes without resulting in cluttering or overlapping. Dense edges also cause severe performance issues in computing graph layout. In Ploceus, the problem of scale is handled using subnetwork sampling. The interface displays bold messages to remind analysts that only a subnetwork is shown. Analysts can interactively add or remove subnetworks through search queries.

A few potential directions exist for future research on this problem. First, it is worthwhile to investigate appropriate mechanisms of subnetwork sampling. Ploceus currently samples randomly. Techniques such as the Degree-Of-Interest functional

retrieval [91] sound more promising. Second, we would like to investigate when it is actually useful to show an overview of the entire network and to articulate the user tasks involved in these situations. It may be possible that we can design alternative visual representations that provide information needed in these tasks without having to show every single node and edge. Thirdly, one way to analyze big graphs is to use a divide and conquer strategy by breaking the graphs down into meaningful subgraphs. Filtering and slicing 'n dicing are two reasonable mechanisms to do so, and they are included in Ploceus. It may be necessary to analyze and compare multiple networks at the same time. Ploceus now organizes multiple networks in the form of a matrix, but there are potential readability and usability problems when each of these networks contains a large number of nodes and edges. While systems like ManyNets [38] have taken a first step in the effort of facilitating visual analysis of multiple networks, more research is needed to understand and to design for multiple-network analysis.

### 6.2.5 Natural Language Interfaces

From the user study described in 5.9, we have identified major hurdles in end-user construction of network visualizations. Learning the operations included in Ploceus and combining these operations to build visualizations still constitute a major challenge for novice users. While training novice users usually helps, we believe there are opportunities to overcome these hurdles through technological means.

One idea we wish to pursue further is the adoption of natural language interfaces. Language is a natural and often effective way for users to articulate what they would like to see and do. For example, instead of having to mentally visualize appropriate visualizations and to construct the visualizations diligently in multiple steps, users could just articulate their intents through words such as "Show me the relationship between researchers and program managers", or "How is Georgia Tech doing in terms of getting grants?". Ideally the system will be able to parse these

natural language queries, either spoken or typed by the users, and then to identify the plausible construction steps using the operations provided in the framework, and to choose appropriate visual encodings to present the final visualizations to the users.

The idea proposed here sounds like a very difficult artificial intelligence (AI) problem. We believe, however, that the problem is actually much more tractable than hard AI problems such as natural language understanding. A major difficulty in this idea is to parse the natural language queries, and this parsing problem is constrained in two ways. First, the vocabulary used often comes from the underlying data space. For example, in "How is Georgia Tech doing in terms of getting grants?", we can identify the nouns without too much difficulty and these nouns presumably are usually data table names, data columns names or specific data cell values. Secondly, the structure of such natural language queries may be mapped to structures in visualizations. For example, in "Show me the relationship between researchers and program managers", the structure in the form of "relationship between X and Y" can be mapped to a restricted set of visualization types by analyzing the data types of X and Y.

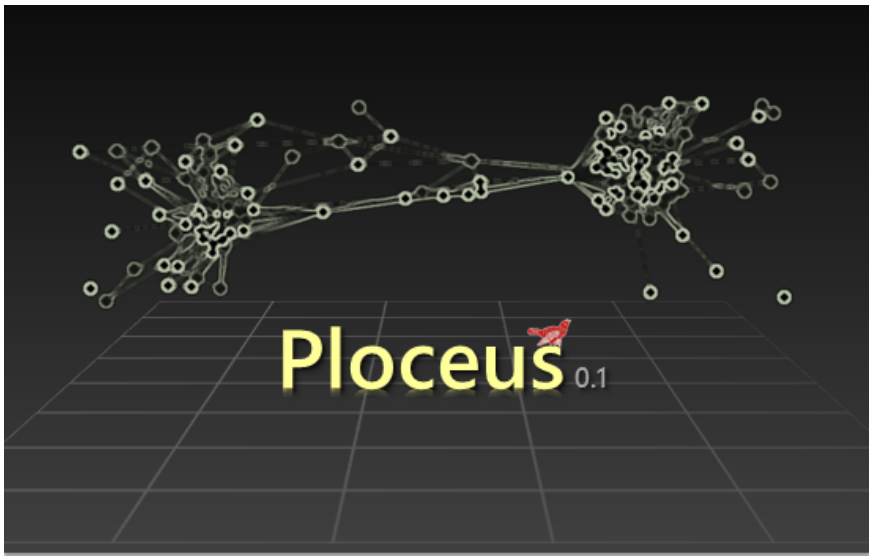
Such a natural language interface does not have to be completely precise in generating the "right" visualization. It can offer a few candidate visualizations that would potentially answer users' questions with varying degrees of confidence. With such an interface, the need for users to mentally visualize and to perform visualization construction is completely eliminated.

# APPENDIX A

# PLOCEUS 0.1 MANUAL

# Ploceus 0.1 Tutorial

Zhicheng Liu
Georgia Institute of Technology




This is a tutorial demonstrating and illustrating features of Ploceus, a visual analytics system. Ploceus supports flexible modeling of tabular data as graphs or networks as well as visual analysis of the modeled networks. This tutorial is in the PDF format, and readers can use the embedded PDF bookmarks or the table of contents on the next page to navigate to different sections of the tutorial.

# Contents

# 1 System Requirements

Ploceus is built completely in Java on top of the Netbeans Rich Client Platform[1]. The system requirements for Ploceus include the following:

- Java 6 or higher
- At least 2GB of memory. By default, Ploceus asks for a maximum of 1GB of RAM. You can change the amount of memory allocated to Ploceus. See Configuring Memory Setting for details.

This is the first release of Ploceus. Any feedback on how to improve the system is greatly appreciated. If you find bugs, have feature requests, or simply want to tell us your experience in working with Ploceus, please send them to zliu6@gatech.edu. Thank you!

# 2 Running Ploceus

### Under Windows

Ploceus is delivered as a zip file. After unzipping, go to the "bin" folder. Double click on "ploceus.exe" to start the system. You can create a shortcut to the .exe file and place it on your desktop.

### Under Mac OS

Ploceus is delivered as a Mac OS application. Double click on "ploceus.app" to launch Ploceus.

# 3 Configuring Memory Settings

Since Ploceus is built in Java, memory available to Ploceus is allocated by Java. If there is too little memory, Ploceus will throw an OutofMemory Exception. On the other hand, if you want to allocate more memory than the system allows, Ploceus will return a "JVM Creation Failed" error.

### Under Windows

To change the memory setting, go to the "etc" folder in the directory where you unzipped Ploceus. Use a text editor to open "ploceus.conf", look for the following line

default_options="--branding ploceus -J-Xms512m -J-Xmx1024m"

Modify the value after the -Xms option and the -Xmx option to change the minimum and maximum heap space. By default the heap size is between 512 and 1024 MB.

### Under Mac OS

Select "Show Package Contents" from the context menu of "ploceis.app". Go to the folder "/ploceus.app/Contents/Resources/ploceus/etc". Use a text editor to open "ploceus.conf", look for the following line

default_options="--branding ploceus -J-Xms512m -J-Xmx1024m"

Modify the value after the -Xms option and the -Xmx option to change the minimum and maximum heap space. By default the heap size is between 512 and 1024 MB.

[1] http://netbeans.org/features/platform/

# 4 System Overview

Ploceus has five window components (Figure 1). The data table column view displays data tables you have imported and the columns within each table. The network schema view shows a schema-level representation of the network you are currently modeling and analyzing (see Section Modeling Networks for details on how to model networks in the schema view). The network visualization view is where visual exploration and analysis take place (see Visualizing and Analyzing Networks for details on the visualization and analysis capabilities Ploceus offers). You can create as many networks as you wish, and each network is shown in a tab. When you switch between the tabs, the network schema view will update accordingly to reflect the schema of the network you are currently examining. The filter panel provides a set of range sliders to filter nodes and edges shown in the visualization. Finally the data table viewer allows you to look at the raw data tables in rows and columns.

Each of these window components can be minimized, maximized and resized. If you have multiple monitors, you can drag a window component out of the main system frame and dock it back using the "Window/Dock Window" menu item. All these features are supported by the Netbeans platform by default.

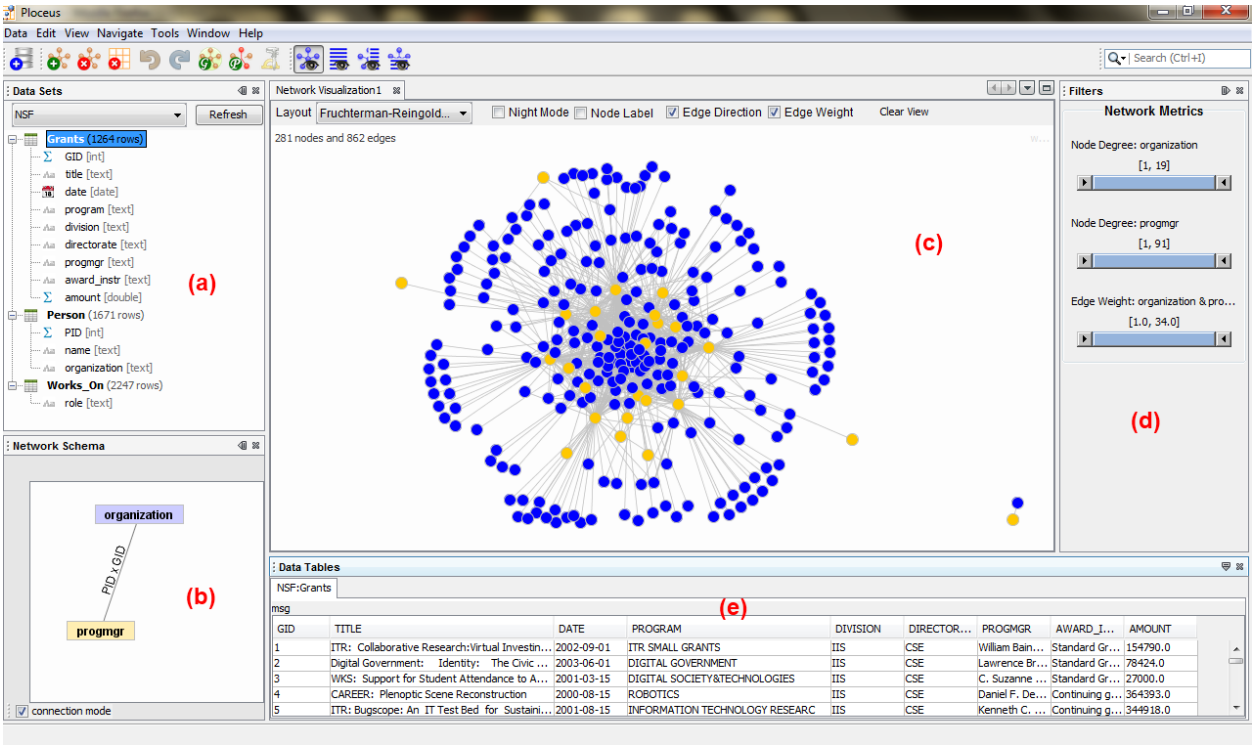



Figure 1: Five window components in the Ploceus system interface: (a) data table column view on the top left, (b) network schema view on the bottom left, (c) network visualization view in the center, (d) filter panel on the right, and (e) data table viewer on the bottom

## 4.1 Managing Networks

In Ploceus, networks that you create are organized as tabs. To create a new blank network, click the "New Network" button (Figure 2) or the "Edit/New Network" menu item. A new tab will open, and you can create networks in this tab using the operations discussed in Modeling Networks.

To delete the networks in the currently active tab, click the "Delete Networks" button (Figure 3) or the "Edit/Delete Networks" menu item. All the nodes and edges will be removed permanently. If you have defined slice 'n dice dimensions (see Slicing 'n Dicing), these dimensions will not be deleted. To delete these dimensions, click the "Delete Slice/Dice Dimensions" button or the "Edit/Delete Slice/Dice Dimensions" menu item.

Whenever you switch between the tabs, the schema view will update to show the schema of the currently active network(s).

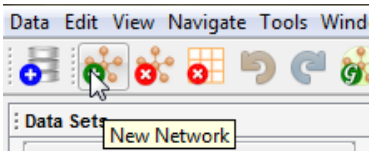



Figure 2: Create an new blank network

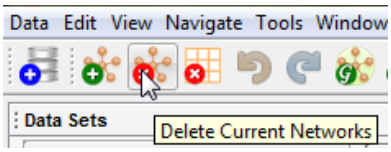



Figure 3: Delete the current network

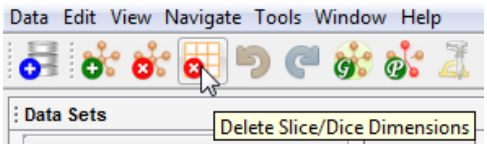



Figure 4: Delete the current slice 'n dice dimensions

# 5 Importing Data

## 5.1 Building Sample Data Sets

Ploceus comes with two sample data sets: white house visitor information from Jan 15, 2009 to Sep 13, 2009, as released on the White House website[2]; and the National Science Foundation research grants awarded in the Information and Intelligent Systems division (thanks to Remco Chang for original scrape of these, and I manually looked up and added researchers' affiliation information) from 2000 to 2003.

To import these sample data sets, simply click on the "Data/Build Sample Data" menu item (Figure 5), and a dialog will pop up showing the progress of data import.

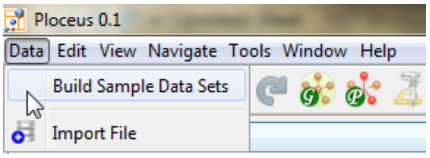



Figure 5: Building Sample Data Sets

## 5.2 Importing Spreadsheets

To import a spreadsheet into Ploceus, click on the "Import File" button in the toolbar, or the "Import File" menu item. A dialog will pop up asking for the location of your file. Currently the Excel file formats (.xls, .xlsx) are supported. After selecting the file to import, Ploceus does an initial processing and shows a dialog for import options (Figure 6).

In version 0.1, only one data sheet can be imported at a time. Each sheet is imported as a separate data source in Ploceus. Once you select the sheet to import, Ploceus will update the dialog to show the top 35 rows of data in the sheet. Ploceus tries to guess the data types of each column and you can change the data type through the combo box located at the top of each column. The header row is highlighted in yellow, and you can specify the header row number. Finally, you need to provide a name for the data source.

[2] http://www.whitehouse.gov/briefing-room/disclosures/visitor-records

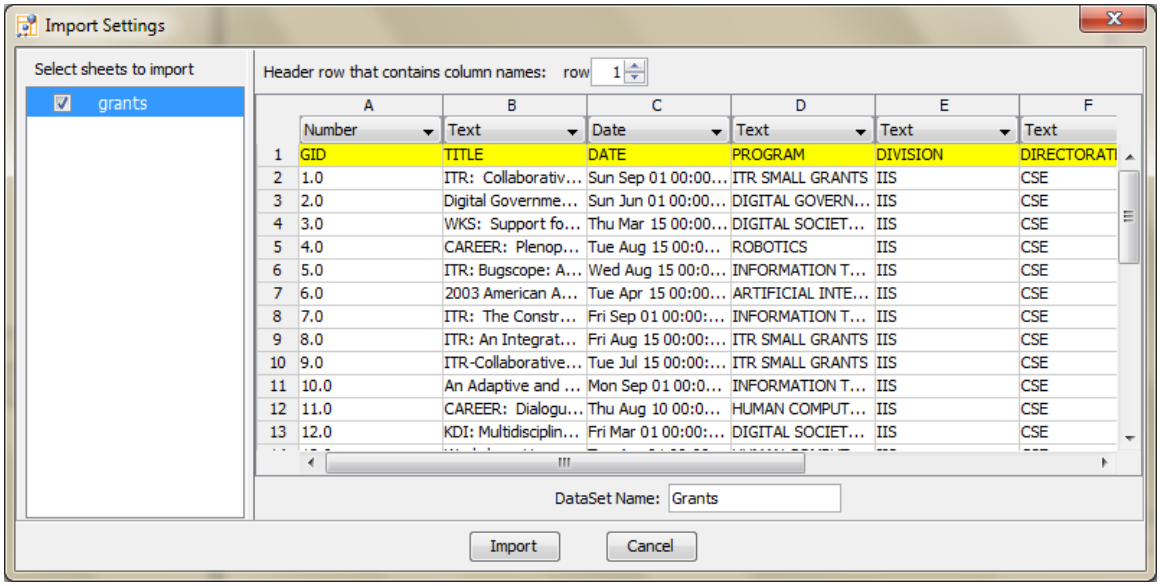



Figure 6: Dialog for Importing Spreadsheet

### 5.3 Importing GraphML files

Support for GraphML file importing will be included in the next release of Ploceus.

### 5.4 Viewing Data Tables

You can view the imported data tables in the data table viewer. If the data table viewer is closed, you can bring it back through the "Window/DataTableViewer" menu item. You can display a data table by dragging and dropping the table name from the data table column view to the data table viewer. Alternatively, you can right click on the table name to bring up a pop-up menu, and click to select "Load Table" (Figure 7).

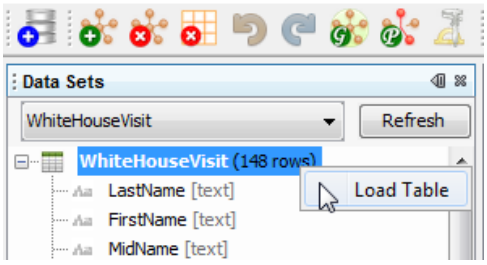



Figure 7: Pop-up menu to display data table

## 6 Modeling Networks

Ploceus supports a variety of network modeling operations. Performing an operation will lead to an automatic update of the visualization in the network visualization view. In this section, we illustrate the modeling operations using a sample data set shown in Table 1, which shows visits to the White House. Each visit is represented as a row in the table. For each visit, it records the last and first name of the person arranging the visit (LastName, FirstName), the type of visit (AccessType), the date (Date) and location (MeetingLoc) of visit, the size of the visiting group (VisitorCount), and the visitee's name (VisiteeLastN, VisiteeFirstN).

Table 1: A table of sample visitor information to the White House

| ID | LastName | FirstName | AccessType | Date | MeetingLoc | VisitorCount | VisiteeLastN | VisiteeFirstN |
|---|---|---|---|---|---|---|---|---|
| 1 | Dodd | Chris | VA | 6/25/09 | WH | 2018 | POTUS | |
| 2 | Smith | John | VA | 6/26/09 | WH | 237 | Office Visitors | |
| 3 | Smith | John | AL | 6/26/09 | OEOB | 144 | Kepko | Amanda |
| 4 | Hirani | Amyn | VA | 6/30/09 | WH | 184 | Office Visitors | |
| 5 | Keehan | Carol | VA | 6/30/09 | WH | 8 | Sheehy | Kristin |
| 6 | Keehan | Carol | VA | 7/8/09 | OEOB | 26 | Leger | Daniella |

### 6.1 Creating Nodes

Transform the values in one or more columns into node labels. For example, you can construct a set of nodes representing the visitors in the White House sample data set, and the labels of the nodes are created from the LastName and FirstName columns. This results in four nodes: “Dodd, Chris”, “Smith, John”, “Hirani, Amyn”, and “Keehan, Carol”.

In the Ploceus interface, you create nodes by dragging selected table columns in the data management view to an empty area in the network schema view. Each drag-and-drop action creates a type of node, and the system assigns a color to that type. (Figure 8).

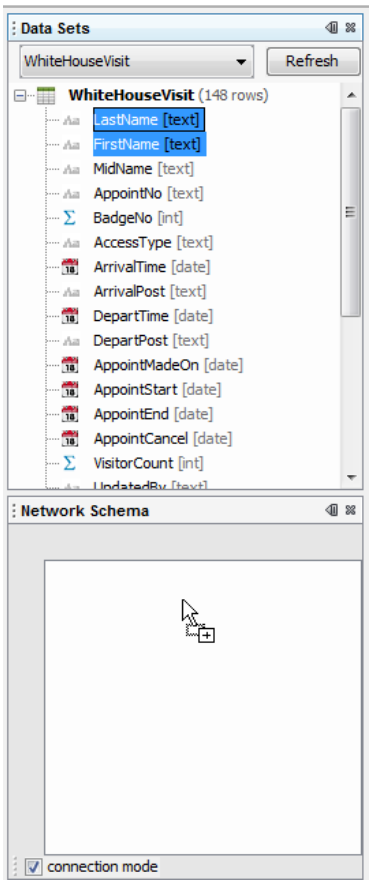



Figure 8: To create nodes, drag selected table columns from the data table column view to an empty area in the network schema view

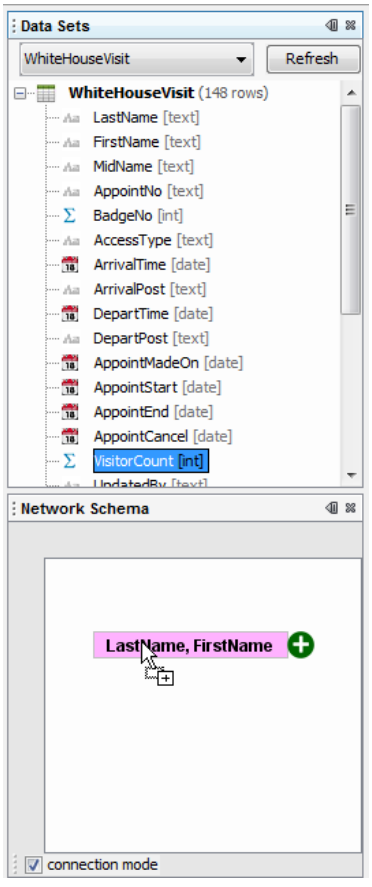



Figure 9: To add attributes, drag selected table columns from the data table column view on top of existing node types in the network schema view

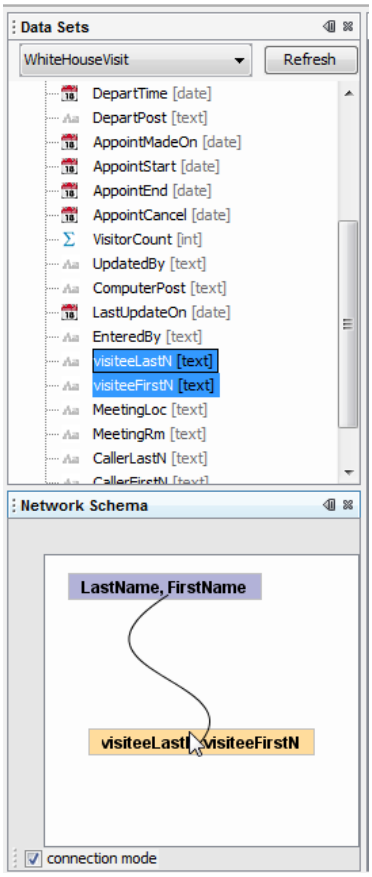



Figure 10: To connect nodes, click on one type of nodes and dragging the mouse to the other type of nodes in the network schema view

### 6.2 Adding Attributes

Transform the values in one or more columns as attributes of existing nodes. Dragging and dropping columns on top of an existing node type add those columns as an attribute to the node type (Figure 9). In Ploceus 0.1, you can only add one attribute for a given node type for the sake of simplicity. If you add an attribute to a node type that already has an attribute, the new attribute will replace the old one.

If you assign a non-quantitative column as attribute, Ploceus allows only one unique attribute value for each node label. For example, if you assign AccessType in Table 1 as an attribute for LastName, FirstName nodes, there will be two "Smith, John" nodes, one having a VA type and the other having an AL type. If you assign a quantitative column as attribute, Ploceus will sum up the quantitative values. For example, you can add an attribute VisitorCount to the people nodes constructed from LastName, FirstName earlier. For the node "Smith, John", its VisitorCount value will be 381.

When the attribute quantitative (e.g. VisitorCount), Ploceus automatically encode the values as node size (i.e. the size of LastName, FirstName nodes represents the value of VisitorCount). When the attribute is categorical (e.g. AccessType), Ploceus automatically encode the attribute values as node color.

### 6.3 Connecting Nodes

Create edges between existing nodes. Given two types of nodes, you create connections between them by clicking on one type of nodes and dragging the mouse to the other type of nodes in the network schema view (Figure 10). This action draws an edge between the two that takes effect when the mouse button is released. Ploceus by default assigns a weight to each edge created, indicating the frequency of co-occurrence between the nodes in the data. For example, if you connect "LastName, FirstName" nodes and "MeetingLoc" nodes from Table 1, by default the edge between "Dodd, Chris" and "WH" has a weight of 1, indicating this person has visited the White House once in this dataset.

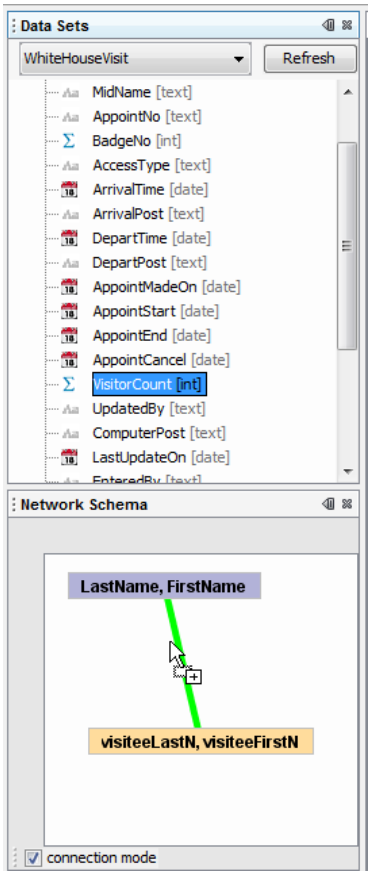



Figure 11: To assign edge weight, drag and drop a quantitative table column over the edge representation in the network schema view

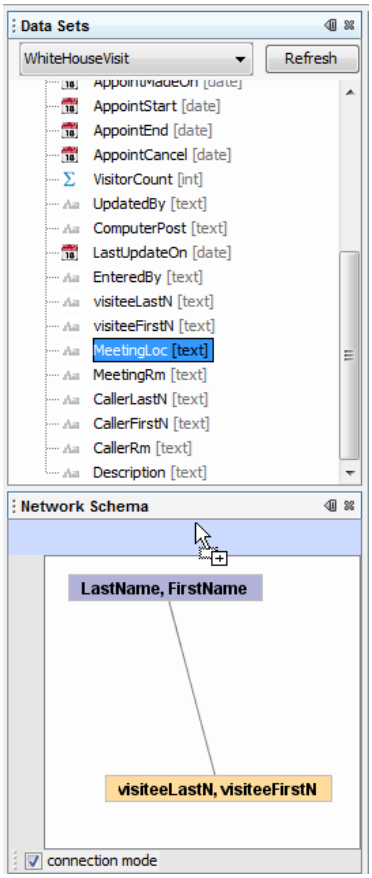



Figure 12: To slice 'n dice, drag and drop a table column to the horizontal or vertical shelf in the network schema view

### 6.4 Assigning Edge Weights

You may want the edge weights to represent something other than the frequency of co-occurrence. To designate a quantitative column as edge weights, drag and drop the column over the edge representation in the network schema view (Figure 11). For example, you can assign the column VisitorCount as the weight for edges connecting "LastName, FirstName" nodes and "MeetingLoc" nodes. The edge between "Dodd, Chris" and "WH" will have a weight of 2018. Only a single column can be assigned as edge weight, and that column must be quantitative.

### 6.5 Slicing 'n Dicing

Divide a network into sub-networks based on selected columns. Ploceus supports slicing and dicing for up to 2 dimensions, designated as the horizontal and vertical axes in the visualization. You specify the orientation of the slices (horizontal or vertical) by dropping columns to the appropriate shelf (Figure 12). A shelf is a horizontal or vertical axis in the network schema view. In Ploceus 0.1, only 1 column is allowed for the horizontal or vertical dimension.

For example, given that we have constructed a LastName, FirstName - VisiteeLastN, VisiteeFirstN network from Table 1, you may want to see how the visiting pattern is related to the locations of visits by dividing the network using MeetingLoc slices. We will then have two subnetworks, one representing the visiting patterns at the White House ("WH"), and the other at the Old Executive Office Building ("OEOB").

### 6.6 Projecting Nodes

Connect two nodes if they both are connected to the same node of a different type. In a two-mode LastName, FirstName - MeetingLoc network, for example, after projecting LastName, FirstName nodes on MeetingLoc nodes, you get a network where visitors are connected to each other if they have visited the same locations, and the weight of the edges reflects the unique number of MeetingLoc shared by two visitors.

In Ploceus, you perform projection through a dialog. The dialog can be invoked from the "Project Network" button in the toolbar (Figure 13) or the "Edit/Project Network" menu item.

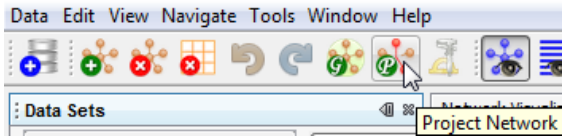



Figure 13: "Project" button in the toolbar

Clicking on the "Project Network" button or menu item brings forth a dialog for you to specify the projection rule through a set of combo boxes (Figure 14). You can preview the result of the projection after specifying the projection rule.

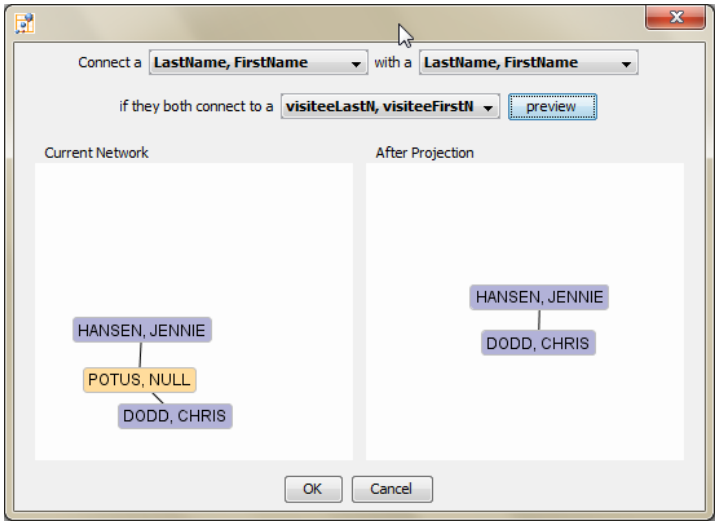



Figure 14: Project dialog in Ploceus

### 6.7 Aggregating Nodes

Group multiple nodes and treat them as one node. As discussed in the Creating Nodes and Adding Attributes sections, Ploceus automatically aggregates nodes with identical labels if no attributes are specified for these nodes, and aggregates nodes with identical labels and values if attributes are specified for the nodes.

In Ploceus, you can define other types of aggregations through a dialog. The dialog can be invoked from the "Aggregate Nodes" button in the toolbar (Figure 15) or the "Edit/Aggregate Nodes" menu item.


Figure 15: "Aggregate Nodes" button in the toolbar

Ploceus 0.1 supports two types of aggregation, made available through two radio buttons (Figure 17).

- *Binning*: divide the range from the minimum to the maximum node values into bins. For example, you can categorize the VisitorCount nodes created from Table 1 into three bins: "small" if VisitorCount $<= 200$, "medium" if $200 <$ VisitorCount $<= 1000$, and "large" if VisitorCount $> 1000$. Binning only works for quantitative values. To create bins, click on the "Create Bins" button in the aggregation dialog (Figure 17), which brings forth a new dialog (Figure 16). In the dialog, the top row shows the minimum and maximum values of the quantitative nodes. On the left, you define the lower and upper bounds of a bin, and gives the bin a unique name. After clicking the "Add Bin" button, the bin is added and all the bins created so far are shown on the right as a list.

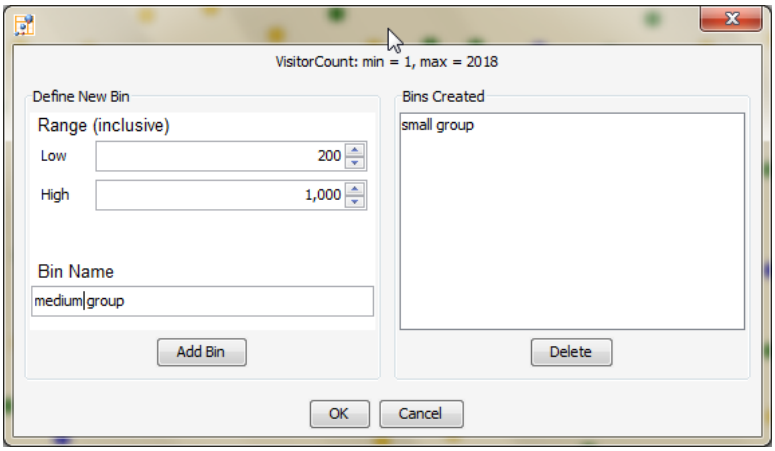



Figure 16: Dialog to create bins for aggregating quantitative nodes

- *Proximity grouping*: group nodes in a pair-wise manner if they have values close to each other. For example, from Table 1 you can create Date nodes from the Date column. You can then aggregate a pair of Date nodes into one if they are within 5 days from each other. This operation is combinatorial: if there are four dates, and every one is within 5 days to each of the other three, proximity grouping will produce $\sum_{k=1}^{(4-1)} k = 6$ nodes. Proximity grouping is useful when combined with projection, so that you can, for example, create a network of visitors whose visiting dates are within 5 days to each other (to do this, connect visitor names with dates, aggregate dates, then project visitors on aggregated dates).

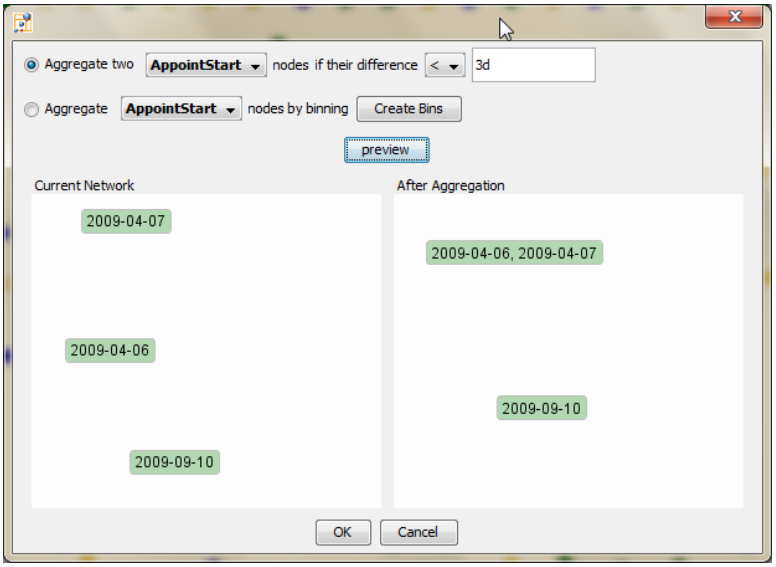



Figure 17: Aggregation dialog in Ploceus with Proximity Grouping Selected

# 7 Visualizing and Analyzing Networks

## 7.1 Choosing Visualizations

Ploceus 0.1 provides two types of visualization: a node-link diagram and a list-based n-partite visualization. You change the type of visualization in the network visualization view through toggle buttons in the toolbar (Figure 18). Four visualization-related buttons are available in the toolbar: node-link, list, tile node-link and list views horizontally, and tile node-link and list views vertically.

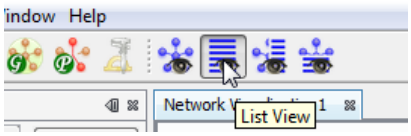



Figure 18: Changing to list-based visualization

In the node-link visualization, five layout algorithms are provided: Fruchterman-Reingold force-directed algorithm [1], circular layout, the Kamada-Kawai algorithm [2], Meyer's self-organizing layout [3], and spring layout.

In the list-based visualization, the nodes in a network are displayed as lists by type. You can sort the nodes within each list alphabetically, or by metrics such as degree centrality, betweenness centrality and closeness centrality. By default, the betweenness and closeness centrality values are not computed because they can be computationally expensive. To compute these values, click the "Compute Analytic Metrics" button in the toolbar or the " Compute Analytic Metrics" menu item (Figure 19).

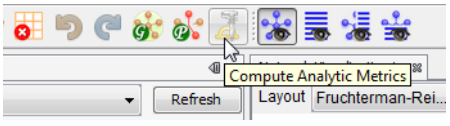



Figure 19: Computing Analytical Metrics

You can show or hide all the edges between two lists of nodes using the checkbox located at the top of the area between two lists. Selecting a node in a list will display edges connecting the nodes to its neighbors.

### 7.2 Handling Big Networks

When the generated network is too big, it is not feasible to display every single node and edges in the node-link visualization. Ploceus 0.1 will display an entire network if its number of nodes is less than 500 and its number of edges is less than 5000, and will randomly sample and display parts of any larger network. Ploceus displays information about the total number of nodes and edges above the node-link visualization (Figure 20).

In the list-based visualization, Ploceus will display all the nodes, organized as lists by type. The edges can be turned on and off using a checkbox.

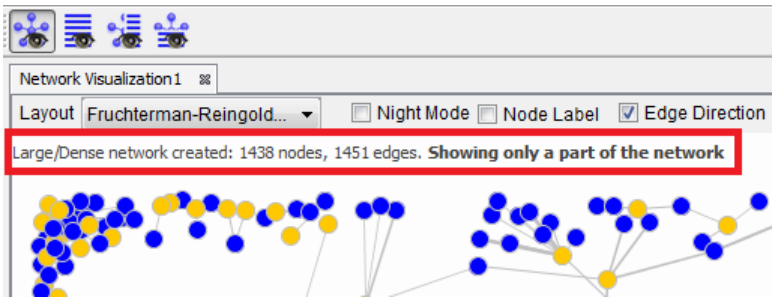



Figure 20: Information about big networks

### 7.3 Searching, Expanding and Hiding Nodes

Ploceus provides interaction mechanisms to query for specific nodes in a big network. The search field is located at the top right corner of the main window (Figure 21). Typing in a search term will result in an instant preview of nodes having labels matching the term. Search terms are not case-sensitive. Pressing "return" will highlight all nodes matching search results. Selecting (by clicking) a single result will highlight that particular node.

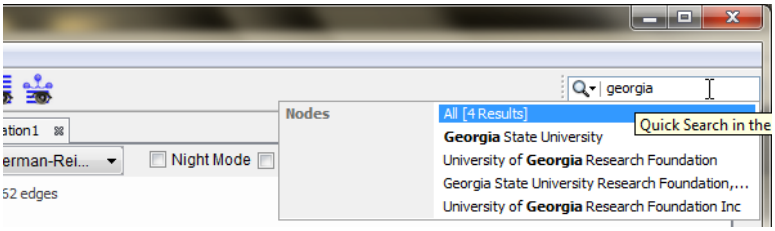



Figure 21: Typing "Georgia" shows search result previews

If a node is not originally shown in the visualization, searching for its label will add the node to the visualization. To see its neighbors, select the node, right click in the background of the node-link view, and select "Expand Selected Nodes". You can also expand a node by double-clicking it.

If you feel that too many nodes are being displayed and the visualization appears noisy, you can remove all the nodes by clicking the "Clear View" button, or select some nodes and right click to bring up the pop-up menu, and choose "Hide Selected Nodes". Note that this action only makes the nodes invisible in the visualization, and is thus different from "delete network" discussed in Managing Networks.

### 7.4 Filtering

You may want to hide certain nodes and edges during visual exploration based on the topological properties of the nodes or edges. Ploceus includes a filtering control panel for you to control the visibility of nodes and edges.

The filtering panel is located on the left of the main window. Just like any other window component in Ploceus, it can be minimized or closed. If it is closed, you can bring it back using the "Window/ControlPanel" menu item.

Ploceus 0.1 includes range sliders for topological properties of node degree and edge weight. A range slider is created for the node degree of each node type and the edge weight of each edge type. The scale of each range slider is discrete

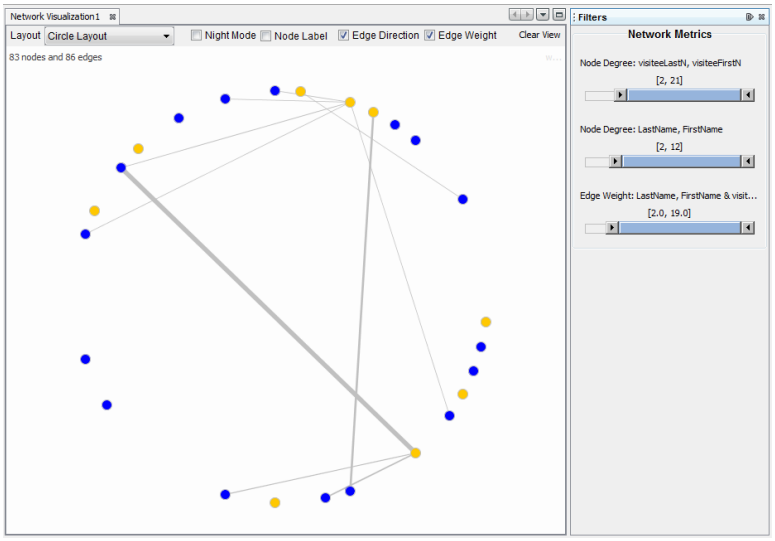



Figure 22: Filtering nodes and edges using the range sliders in the filtering panel

instead of continuous: for a node type with a degree distribution of $\{1, 2, 5, 20\}$, for example, there will be four values, instead of twenty values (i.e. 1-20), assigned to the range slider.

# 8 FAQ

## 8.1 What do the edges mean?

The edges are constructed based on row-based co-occurrences of values in the original data table. For example, if you connect the MeetingLoc and the AccessType columns in Table 1, ask yourself: what does it mean when a meeting location appears in the same row with an access type? The edge weight will reflect the frequency of row-based co-occurrences.

## 8.2 More FAQs to come later

More FAQs to come later...

# References

[1] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software- Practice and Experience*, 21(11):1129–1164, 1991.

[2] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989.

[3] B. Meyer. Self-organizing graphs - a neural network perspective of graph layout. In *Graph Drawing*, pages 246–262, 1998.

# REFERENCES


[1] "Gephi, an open source graph visualization and manipulation software." http://gephi.org, Aug. 2010.

[2] "GraphML file format." http://graphml.graphdrawing.org/, Aug. 2010.

[3] "NetMiner - social network analysis software." http://www.netminer.com, Aug. 2010.

[4] "OLAP and OLAP server definitions." http://www.olapcouncil.org/research/glossaryly.htm, Sept. 2010.

[5] "Pajek - program for large network analysis." http://pajek.imfm.si/doku.php, Aug. 2010.

[6] "Tableau software." http://www.tableausoftware.com/, Aug. 2010.

[7] "UCINET." http://www.analytictech.com/ucinet/, Aug. 2010.

[8] "Centrifuge systems." http://centrifugesystems.com/, June 2011.

[9] "Excel DataScope." http://research.microsoft.com/en-us/projects/azure/datascope.aspx, July 2011.

[10] "H2 database engine." http://www.h2database.com/html/main.html, Mar. 2011.

[11] "NetBeans Rich-Client platform development." http://netbeans.org/features/platform/, Mar. 2011.

[12] "TouchGraph navigator: Graph visualization and social network analysis software." http://touchgraph.com/navigator, Mar. 2011.

[13] Adar, E., "GUESS: a language and interface for graph exploration," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, 2006.

[14] Amar, R. A. and Stasko, J. T., "Knowledge precepts for design and evaluation of information visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 4, pp. 432–442, 2005.

[15] Auber, D., "Tulip: a huge graph visualization framework," *Graph Drawing Software, Mathematics and Visualization*, 2003.

[16] Bagui, S. and Earp, R., *Database Design Using Entity-Relationship Diagrams*. Auerbach Publications, 1 ed., 2003.

[17] Bendix, F., Kosara, R., and Hauser, H., "Parallel sets: Visual analysis of categorical data," in *Proceedings of the IEEE Symposium on Information Visualization*, pp. 133âĂŞ–140, 2005.

[18] Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., and Wiswedel, B., "KNIME: the konstanz information miner," *Data Analysis, Machine Learning and Applications*, pp. 319–326, 2008.

[19] Bertin, J., *Semiology of Graphics*. The University of Wisconsin Press, 1983.

[20] Bhattacharya, I. and Getoor, L., "Collective entity resolution in relational data," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 5, 2007.

[21] Bishop, Y. M., Fienberg, S. E., and Holland, P. W., *Discrete Multivariate Analysis: Theory and Practice*. Springer, 1 ed., July 2007.

[22] Bohannon, P., Korth, H. F., and Narayan, P. P. S., "The table and the tree: On-line access to relational data through virtual XML documents," in *Proc. of WebDB*, 2001.

[23] Chaudhuri, S. and Dayal, U., "An overview of data warehousing and OLAP technology," *ACM SIGMOD Record*, vol. 26, pp. 65–74, Mar. 1997. ACM ID: 248616.

[24] Chen, C., Yan, X., Zhu, F., Han, J., and Yu, P. S., "Graph OLAP: a multi-dimensional framework for graph data analysis," *Knowledge and Information Systems*, vol. 21, pp. 41–63, Oct. 2009. ACM ID: 1669197.

[25] Chen, P. P., "The entity-relationship model – toward a unified view of data," *ACM Transactions on Database Systems*, vol. 1, no. 1, pp. 9–36, 1976.

[26] Christensen, R., *Log-Linear Models and Logistic Regression*. Springer, 2nd ed., Sept. 1997.

[27] Cleveland, W. S. and McGill, R., "Graphical perception and graphical methods for analyzing scientific data," *Science*, vol. 229, no. 4716, pp. 828–833, 1985.

[28] Codd, E. F., "A relational model of data for large shared data banks," *Communications of the ACM*, vol. 13, no. 6, p. 387, 1970.

[29] Cypher, A. and Halbert, D. C., *Watch what I do: programming by demonstration*. The MIT Press, 1993.

[30] Derthick, M., Kolojejchick, J., and Roth, S. F., "An interactive visualization environment for data exploration," *IN PROC. OF KNOWLEDGE DISCOVERY IN DATABASES*, pp. 2—9, 1997.

[31] Dijkstra, E. W., "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[32] Dunne, C. and Shneiderman, B., "Improving graph drawing readability by incorporating readability metrics: A software tool for network analysts," *University of Maryland, HCIL Tech Report HCIL-2009-13*, 2009.

[33] Ehrig, H., "Introduction to the algebraic theory of graph grammars (A survey)," in *Proceedings of the International Workshop on Graph-Grammars and Their Application to Computer Science and Biology*, pp. 1–69, Springer-Verlag, 1979. ACM ID: 730053.

[34] Elmasri, R. and Navathe, S., *Fundamentals of Database Systems*. Addison Wesley, 6th ed., 2011.

[35] Faloutsos, C., McCurley, K. S., and Tomkins, A., "Fast discovery of connection subgraphs," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 118–127, 2004.

[36] Faloutsos, M., Faloutsos, P., and Faloutsos, C., "On power-law relationships of the internet topology," in *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '99, (Cambridge, Massachusetts, United States), pp. 251–262, ACM, 1999. ACM ID: 316229.

[37] Freeman, L. C., "Centrality in social networks conceptual clarification," *Social networks*, vol. 1, no. 3, pp. 215–239, 1979.

[38] Freire, M., Plaisant, C., Shneiderman, B., and Golbeck, J., "ManyNets: an interface for multiple network analysis and visualization," in *Proceedings of the 28th international conference on Human factors in computing systems*, pp. 213–222, 2010.

[39] Fruchterman, T. M. and Reingold, E. M., "Graph drawing by force-directed placement," *Software- Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

[40] Getoor, L. and Diehl, C. P., "Link mining: a survey," *SIGKDD Explor. Newsl.*, vol. 7, pp. 3–12, Dec. 2005. ACM ID: 1117456.

[41] Ghoniem, M., Fekete, J., and Castagliola, P., "A comparison of the readability of graphs using Node-Link and Matrix-Based representations," in *Proceedings of the IEEE Symposium on Information Visualization*, pp. 17–24, IEEE Computer Society, 2004.

[42] Görg, C., Kihm, J., Choo, J., Liu, Z., Muthiah, S., Park, H., and Stasko, J., "Combining computational analyses and interactive visualization to enhance information retrieval," in *Fourth Workshop on Human-Computer Interaction and Information Retrieval*, 2010.

[43] Grammel, L., Tory, M., and Storey, M., "How information visualization novices construct visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, pp. 943–952, Dec. 2010.

[44] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., and Pirahesh, H., "Data cube: A relational aggregation operator generalizing Group-By, Cross-Tab, and Sub-Totals," *Data Mining and Knowledge Discovery*, vol. 1, pp. 29–53, Mar. 1997.

[45] Guéhis, S., Rigaux, P., and Waller, E., "Data-driven publication of relational databases," in *10th International Database Engineering and Applications Symposium*, pp. 267–272, 2006.

[46] Han, J., Kamber, M., and Pei, J., *Data Mining: Concepts and Techniques*. Elsevier, June 2011.

[47] Hansen, D., Shneiderman, B., and Smith, M. A., *Analyzing Social Media Networks with NodeXL: Insights from a Connected World*. Morgan Kaufmann, Sept. 2010.

[48] Heer, J., Card, S. K., and Landay, J. A., "prefuse: a toolkit for interactive information visualization," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 421–430, 2005.

[49] Heer, J. and Perer, A., "Orion: A system for modeling, transformation and visualization of multidimensional heterogeneous networks," in *IEEE Conference on Visual Analytics Science and Technology '11*, 2011.

[50] Henry, N., Fekete, J., and McGuffin, M. J., "NodeTrix: a hybrid visualization of social networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1302–1309, 2007.

[51] Hey, A. J., Tansley, S., and Tolle, K. M., *The fourth paradigm: data-intensive scientific discovery*. Microsoft Research Redmond, WA, 2009.

[52] Inselberg, A., "Multidimensional detective," in *IEEE Symposium on Information Visualization*, pp. 100–107, 1997.

[53] Jamali, M. and Abolhassani, H., "Different aspects of social network analysis," in *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 66–72, 2006.

[54] Jelen, B. and Alexander, M., *Pivot Table Data Crunching*. Que, July 2005.

[55] Kamada, T. and Kawai, S., "An algorithm for drawing general undirected graphs," *Information processing letters*, vol. 31, no. 1, pp. 7–15, 1989.

[56] Keim, D. A., Hao, M. C., Dayal, U., and Hsu, M., "Pixel bar charts: a visualization technique for very large multi-attribute data sets," *Information Visualization*, vol. 1, no. 1, pp. 20–34, 2002.

[57] Keim, D. A. and Kriegel, H. P., "VisDB: database exploration using multidimensional visualization," *IEEE Computer Graphics and Applications*, vol. 14, pp. 40–49, Sept. 1994.

[58] Knoke, D. and Yang, S., *Social Network Analysis*. Sage Publications, Inc, 2nd ed., Nov. 2007.

[59] Kobsa, A., "An empirical comparison of three commercial information visualization systems," in *IEEE Symposium on Information Visualization*, pp. 123–130, 2001.

[60] Krasner, G. and Pope, S., "A description of the model-view-controller user interface paradigm in the smalltalk-80 system," *Journal of object oriented programming*, vol. 1, no. 3, pp. 26–49, 1988.

[61] Latapy, M., Magnien, C., and Vecchio, N. D., "Basic notions for the analysis of large two-mode networks," *Social Networks*, vol. 30, no. 1, pp. 31–48, 2008.

[62] Lee, B., Plaisant, C., Parr, C. S., Fekete, J., and Henry, N., "Task taxonomy for graph visualization," in *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, (Venice, Italy), pp. 1–5, ACM, 2006.

[63] Liu, B. and Jagadish, H. V., "A spreadsheet algebra for a direct data manipulation query interface," in *Proceedings of the 2009 IEEE International Conference on Data Engineering*, pp. 417–428, 2009.

[64] Liu, Z., Stasko, J., and Sullivan, T., "SellTrend: Inter-Attribute visual analysis of temporal transaction data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1025–1032, 2009.

[65] Liu, Z. and Stasko, J., "Mental models, visual reasoning and interaction in information visualization: A top-down perspective," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, 2010.

[66] Livny, M., Ramakrishnan, R., Beyer, K., Chen, G., Donjerkovic, D., Lawande, S., Myllymaki, J., and Wenger, K., "DEVise: integrated querying and visual exploration of large datasets," *SIGMOD Rec.*, vol. 26, pp. 301–312, June 1997. ACM ID: 253335.

[67] Loscalzo, S. and Yu, L., "Social network analysis: Tasks and tools," *Social Computing, Behavioral Modeling, and Prediction*, pp. 151–159, 2008.

[68] Mackinlay, J., "Automating the design of graphical presentations of relational information," *ACM Transactions on Graphics*, vol. 5, no. 2, pp. 110–141, 1986.

[69] Mackinlay, J. D., Hanrahan, P., and Stolte, C., "Show me: Automatic presentation for visual analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, pp. 1137–1144, Dec. 2007.

[70] MacQueen, J. and others, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, p. 14, 1967.

[71] Meijer, E. and Bierman, G., "A co-Relational model of data for large shared data banks," *Queue*, vol. 9, pp. 30–48, Mar. 2011. ACM ID: 1961297.

[72] Meyer, B., "Self-organizing graphs - a neural network perspective of graph layout," in *Graph Drawing*, pp. 246–262, 1998.

[73] O'Madadhain, J., Fisher, D., White, S., and Boey, Y., "JUNG: Java Universal Network/Graph Framework," Tech. Rep. Technical Report UCI-ICS 03-17, 2003.

[74] Perer, A. and Shneiderman, B., "Balancing systematic and flexible exploration of social networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 693–700, 2006.

[75] Pirolli, P. and Rao, R., "Table lens as a tool for making sense of data," in *Proceedings of the workshop on Advanced visual interfaces*, pp. 67–80, 1996.

[76] Rao, R. and Card, S. K., "The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information," in *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*, (Boston, Massachusetts, United States), pp. 318–322, ACM, 1994.

[77] Shneiderman, B. and Aris, A., "Network visualization by semantic substrates," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 733–740, 2006.

[78] Shneiderman, B. and Plaisant, C., "Strategies for evaluating information visualization tools: Multi-dimensional in-depth long-term case studies," in *BELIV : BEyond time and errors: novel evaLuation methods for Information Visualization*, (Venice, Italy), 2006.

[79] Smith, M. A., Shneiderman, B., Milic-Frayling, N., Rodrigues, E. M., Barash, V., Dunne, C., Capone, T., Perer, A., and Gleave, E., "Analyzing (social media) networks with NodeXL," in *Proceedings of the fourth international conference on Communities and technologies*, (University Park, PA, USA), pp. 255–264, ACM, 2009.

[80] Spenke, M. and Beilken, C., "InfoZoom: Analysing Formula One racing results with an interactive data mining and visualisation tool," in *Proceedings of 2nd International Conference on Data Mining*, (Cambridge University, UK), pp. 455–464, 2000.

[81] Spenke, M., Beilken, C., and Berlage, T., "FOCUS: the interactive table for product comparison and selection," in *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pp. 41–50, 1996.

[82] Stasko, J., Görg, C., and Liu, Z., "Jigsaw: supporting investigative analysis through interactive visualization," *Information Visualization*, vol. 7, no. 2, pp. 118–132, 2008.

[83] Stasko, J., Görg, C., and Liu, Z., "Sensemaking across text documents: human-centered, visual exploration with jigsaw," in *Sensemaking Workshop, CHI 2008*, 2008.

[84] Stolte, C., Tang, D., and Hanrahan, P., "Polaris: A system for query, analysis, and visualization of multidimensional relational databases," *IEEE Transactions on Visualization and Computer Graphics*, pp. 52–65, 2002.

[85] Stolte, C., Tang, D., and Hanrahan, P., "Multiscale visualization using data cubes," *IEEE Transactions on Visualization and Computer Graphics*, pp. 176–187, 2003.

[86] Stolte, C., Tang, D., and Hanrahan, P., "Query, analysis, and visualization of hierarchically structured data using polaris," *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 112–122, 2002. ACM ID: 775064.

[87] Stolte, C. R., *Query, analysis, and visualization of multidimensional databases*. PhD thesis, Stanford University, 2003.

[88] Thomas, J. J. and Cook, K. A., "Illuminating the path: The research and development agenda for visual analytics," *IEEE Computer Society*, 2005.

[89] Tian, Y., Hankins, R. A., and Patel, J. M., "Efficient aggregation for graph summarization," *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 567–580, 2008. ACM ID: 1376675.

[90] Tukey, J. W., *Exploratory data analysis*. Addison-Wesley, 1977.

[91] van Ham, F. and Perer, A., ""Search, show context, expand on demand": Supporting large graph exploration with Degree-of-Interest," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, pp. 953–960, Nov. 2009. ACM ID: 1639191.

[92] Ward, M. O., "XmdvTool: integrating multiple methods for visualizing multivariate data," in *IEEE Conference on Visualization, 1994., Visualization '94, Proceedings*, pp. 326–333, IEEE, Oct. 1994.

[93] Ware, C., *Information Visualization: Perception for Design*. Morgan Kaufmann, 2004.

[94] Ware, C., *Visual Thinking for Design*. Morgan Kaufmann, 2008.

[95] Wasserman, S., *Social network analysis: Methods and applications*. Cambridge university press, 1994.

[96] Wattenberg, M., "Visual exploration of multivariate graphs," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 811–819, ACM New York, NY, USA, 2006.

[97] Weaver, C., "Multidimensional data dissection using attribute relationship graphs," in *Visual Analytics Science and Technology (VAST), 2010 IEEE Symposium on*, pp. 75–82, 2010.

[98] Wilkinson, L., *The Grammar of Graphics*. Springer Verlag, 2005.

[99] Witkowski, A., Bellamkonda, S., Bozkaya, T., Naimat, A., Sheng, L., Subramanian, S., and Waingold, A., "Query by excel," in *Proceedings of the 31st international conference on Very large data bases*, p. 1215, 2005.

[100] Yu, P. S., Han, J., and Faloutsos, C., *Link Mining: Models, Algorithms, and Applications*. Springer, 1st edition. ed., Sept. 2010.

[101] Zhao, P., Li, X., Xin, D., and Han, J., "Graph cube: On warehousing and OLAP multidimensional networks," in *Proc. of 2011 ACM SIGMOD Int. Conf. on Management of Data*, 2011.